ePOS-Print SDK for Android

User's Manual

EPSON®
EXCEED YOUR VISION

Overview

Describes the features and development environment.

Sample Program

Describes how to use the sample program.

Programming Guide

Describes how to write programs in application development.

API Reference

Describes the APIs provided in ePOS-Print SDK for Android.

Command Transmission/Reception

Describes the APIs for transmitting and receiving commands.

Appendix

Describes the specifications for printers used for the ePOS-Print SDK for Android.

Cautions

- No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Seiko Epson Corporation.

- The contents of this document are subject to change without notice. Please contact us for the latest information.

- While every precaution has taken in the preparation of this document, Seiko Epson Corporation assumes no responsibility for errors or omissions.

- Neither is any liability assumed for damages resulting from the use of the information contained herein.

- Neither Seiko Epson Corporation nor its affiliates shall be liable to the purchaser of this product or third parties for damages, losses, costs, or expenses incurred by the purchaser or third parties as a result of: accident, misuse, or abuse of this product or unauthorized modifications, repairs, or alterations to this product, or (excluding the U.S.) failure to strictly comply with Seiko Epson Corporation's operating and maintenance instructions.

- Seiko Epson Corporation shall not be liable against any damages or problems arising from the use of any options or any consumable products other than those designated as Original EPSON Products or EPSON Approved Products by Seiko Epson Corporation.

Trademarks

EPSON® and ESC/POS® are registered trademarks of Seiko Epson Corporation in the U.S. and other countries.

Android™ is either registered trademarks or trademarks of Google Inc. in the United States and other countries.

Java™ is a registered trademark of Oracle Corporation, its subsidiaries, and affiliates in the U.S. and other countries.

Wi-Fi® is a registered trademark of the Wi-Fi Alliance®.

Bluetooth® is a registered trademark of Bluetooth SIG, Inc.

Eclipse® is a trademark or registered trademark of Eclipse Foundation, Inc.

ESC/POS® Command System

EPSON has been taking industry's initiatives with its own POS printer command system (ESC/POS). ESC/POS has a large number of commands including patented ones. Its high scalability enables users to build versatile POS systems. The system is compatible with all types of EPSON POS printers (excluding the TM-C100) and displays. Moreover, its flexibility makes it easy to upgrade the future. The functionality and the user-friendliness is valued around the world.

For Safety

Key to Symbols

The symbols in this manual are identified by their level of importance, as defined below. Read the following carefully before handling the product.

⚠	Provides information that must be observed to avoid damage to your equipment or a malfunction.
✎	Provides important information and useful tips.

Restriction of Use

When this product is used for applications requiring high reliability/safety such as transportation devices related to aviation, rail, marine, automotive etc.; disaster prevention devices; various safety devices etc; or functional/precision devices etc, you should use this product only after giving consideration to including fail-safes and redundancies into your design to maintain safety and total system reliability. Because this product was not intended for use in applications requiring extremely high reliability/safety such as aerospace equipment, main communication equipment, nuclear power control equipment, or medical equipment related to direct medical care etc, please make your own judgment on this product's suitability after a full evaluation.

About this Manual

Aim of the Manual

This manual aims to provide development engineers with all the information necessary for the construction and design of a printing system that uses ePOS-Print SDK, and for the development and design of printer applications.

Manual Content

The manual is made up of the following sections:

Contents

Command Transmission/Reception... 155

Appendix 165

Overview

This chapter describes the features of and the specifications for ePOS-Print SDK for Android.

Overview of ePOS-Print SDK

The ePOS-Print SDK for Android is an SDK aimed at development engineers who are developing Android applications for printing on an EPSON TM printer. Applications are developed using the APIs provided by ePOS-Print SDK.

The ePOS-Print SDK also has the "ePOS-Print SDK for iOS" for iOS applications.

> APIs for transmitting/receiving commands to/from TM printers are also provided.
> A command transmission/reception API cannot be used with the ePOS-Print API, Print class. For details on the command transmission/reception APIs, refer to Command Transmission/Reception (p.155).

Features

❏   Allows printing to TM printers from Android applications.

❏   Allows acquisition of TM printer status from Android applications.

Function

ePOS-Print API

❏ Print setting (alignment/line feed space/text rotation/page mode)

❏ Character data setting (language/font (device font)/double-sizing/scale/smoothing/print position)

❏ Character style setting (inversion of black and white/underline/bold)

❏ Paper feed setting (in dots/in lines)

❏ Image printing (raster image/NV graphics)

❏ Barcode printing
(For barcodes that can be printed by each model, refer to Printer specifications (p.165).)

❏ 2D-code printing
(For 2D-code that can be printed by each model, refer to Printer specifications (p.165).)

❏ Drawer kick function

❏ Buzzer function

❏ Paper layout setting

❏ Label / black mark paper feed setting

❏ ESC/POS command transmission

❏ Acquisition of response from printer (printing result / printer status / battery status)

❏ Compatible with Asian languages (simplified Chinese, traditional Chinese, Korean, Thai, Vietnamese)

Printer Search API

❏ Search for printers

Log Setting API

❏ Log output setting
(This API allows to output log data to an Android device's storage and a server that can establish TCP connection.)

> Log data output to an Android device can be saved on other computers using a USB connection.

Operating Environment

Android Version

❏ Android Version 2.3.3 to 2.3.7

❏ Android Version 3.1 to 3.2.2

❏ Android Version 4.0 to 4.2.2

> For the latest version, refer to the README file.

Android Device

Device that supports ARMv5TE

Printer

TM Printer	Interface		
	Wired LAN	Wi-Fi	Bluetooth
TM-T88V	✔	✔	-
TM-T20	✔	-	-
TM-T70	✔	✔	-
TM-T70II	✔	✔	-
TM-T82	✔	-	-
TM-T82II	✔	-	-
TM-T81II	✔	-	-
TM-T90II	✔	✔	-
TM-P60(Receipt) Wi-Fi	-	✔	-
TM-P60(Receipt) Bluetooth	-	-	✔
TM-P60(Peeler) Wi-Fi	-	✔	-
TM-P60(Peeler) Bluetooth	-	-	✔
TM-P60II(Receipt) Wi-Fi	-	✔	-
TM-P60II(Receipt) Bluetooth	-	-	✔
TM-P60II(Peeler) Wi-Fi	-	✔	-
TM-P60II(Peeler) Bluetooth	-	-	✔
TM-U220 Series	✔	✔	-

Development Environment

The following are necessary to develop an Android application.

- Android SDK r15 or later
- Java Development Kit 6 or later

Contents in the Package

Package

File	Description
ePOS-Print.jar	Compiled Java class file, archived into a jar format file to allow APIs to be used from Java programs.
libeposprint.so	Library for function execution. (ARMv5TE supported)
ePOS-Print_Sample_Android.zip	A sample program file.
README.en.txt	A readme file.
README.jp.txt	A readme file. (The Japanese-language edition)
EULA.en.txt	Contains the SOFTWARE LICENSE AGREEMENT.
EULA.jp.txt	Contains the SOFTWARE LICENSE AGREEMENT. (The Japanese-language edition)
ePOS-Print_SDK_Android_E_Revx.pdf	This manual.
ePOS-Print_SDK_Android_J_Revx.pdf	The Japanese-language edition of this manual.

Manual

The following manuals are available for ePOS-Print SDK for Android.
❏   ePOS-Print SDK for Android User's Manual (This Document)

❏   ePOS-Print SDK for Android Application Development - Setup Guide

Sample Program

For an Android application for TM printers developed using ePOS Print SDK for Android, the following program is available.
❏   ePOS-Print_Sample_Android.zip

Download

For customers in North America, go to the following web site:
     http://www.epsonexpert.com/
For customers in other countries, go to the following web site:
     https://download.epson-biz.com/?service=pos

Restrictions

❏ A communication API (p.47) and command transmission/reception API (p.155) in the ePOS Print APIs cannot be used for the same device at the same time.

❏ A maximum of 16 device ports can be opened in the same application at the same time.

❏ More than one port cannot be opened for the same device at the same time.

❏ When the screen display rotates, Activity may be discarded. To retain a Print instance using Activity, closePrinter of the Print class should be called before Activity is discarded.

❏ If the device goes into sleep mode while communicating with a printer via Bluetooth, the connection will be lost.

Sample Program

This chapter describes how to use the sample program(ePOS-Print Sample Program for Android).

> • The sample program is provided as an Android application project for use with Eclipse, including the Java source files.
> • For an Android application for TM printers developed using ePOS Print SDK, the following program is available.

Functionality

The Sample Program has the following functionality.

❏ Searching for printers

❏ Opening of port

❏ Closing of port

❏ Text printing

❏ Graphic printing (image file printing)

❏ Barcode printing

❏ 2D-code printing

❏ Printing in page mode

❏ Paper cutting

❏ Printer status acquisition

❏ Acquisition of printer model name/language information

❏ Log output setting

❏ Display of status event

❏ Display of battery status event

> The sample program does not contain a functionality for turning text / images / barcodes / etc.

15

Usage Environment

Development Environment

- Android SDK r16
- Java Development Kit 6
- Eclipse
- ADT Plugin for Eclipse

> For details about ways to construct a development environment, please refer to the "ePOS-Print SDK for Android Application Development - Setup Guide".

Printer

- TM printer supported in ePOS-Print SDK.

Target device

- Device connected to a computer via USB

Environmental Construction

Follow the procedures below to use the sample program.

1 Extract the sample program zip file to a directory of your choosing.

2 In Eclipse, go to (File)-(Import), select (General)-(Existing Project into Workspace), and then click (Next).

3 The Import Projects window will be displayed. Make the settings shown below and click (Finish).

Item	Setting
Select root directory	Specify the directory where you extracted the sample program zip file.
Copy projects into workspace	Check this option.

4 In Package Explorer view, right click on the "ePOSPrintSample" project and select (Properties).

5 The Properties for ePOSPrintSample window will be displayed. Make the settings shown below and click (OK).

Item	Setting
Android	Select the Android OS version of the target device.
Java Build Path	In Libraries, confirm that the path to ePOS-Print.jar, which is located in the ePOSPrintSample project that was copied into the project workspace, is set.

6 In Package Explorer view, right click on the "ePOSPrintSample" project and select (Run As, Android Application).

7 The sample program will be installed to the target Android device, and then the program will start up.

How to Use the Program Sample

This section describes how to use the program sample for the following operations:

- Search for printers and printing (p.18)
- Acquisition of Printer Model Name (p.25)

Search for printers and printing

Use the sample program as follows:

1 Start the sample program. For details, refer to Environmental Construction (p.17).

2 Search for printers. Tap (Printer Discovery) on the main screen.
When you select (Device Type), the IP addresses/Mac addresses for the detected printers are listed.

3 Select the IP address/Mac address of the printer you want to use from the list of IP addresses/Mac address displayed in procedure 2.

4 Open the printer's port. Tap (Open) on the main screen.
The "Device Type" and "IP Address/Mac Address" of the printer selected in procedure 3 are displayed. Select (Printer Name) and (Language).

5 Set (Status Monitor).

Item	Description
Enabled	• ON: The status monitor is enabled and the printer status is monitored. • OFF: The status monitor is disabled.
Interval	When Enabled is turned ON, the status monitoring interval is set in units of milliseconds.

6 Tap (Open).

7 Execute the following processes:

Process	Description
Text printing	Tap (Text) on the main screen. For details, refer to Text printing (p.20).
Graphic printing	Tap (Image) on the main screen. For details, refer to Graphic printing (p.20).
Barcode printing	Tap (Barcode) on the main screen. For details, refer to Barcode printing (p.21).
2D-code printing	Tap (2D Code) on the main screen. For details, refer to 2D-code printing (p.21).
Printing in page mode	Tap (Page Mode) on the main screen. For details, refer to Printing in page mode (p.22).
Paper cutting	Tap (Cut) on the main screen. For details, refer to Paper cutting (p.22).
Log output setting	Tap (Log Settings) on the main screen. For details, refer to Log output setting (p.22).
Printer status acquisition	Tap (Get Status) on the main screen.

8 The following execution results will be displayed:
- Process execution result (error status / printer status / battery status)
  For details, refer to Process execution result (p.23).
- Method (API) execution error
  For details, refer to Method (API) execution error (p.24).

9 When all processing is finished, tap (Close) on the main screen, and close the printer's port.

2

Text printing

Execute the text printing according to the following procedure:

1 Enter a string to print for (Print Characters).

2 Specifies the character properties for the string to print. The following properties can be specified:

Property	Description
Font	Set the character font.
Align	Set the alignment.
Line Spacing	Set the line feed space.
Language	Set the language.
Size	Set the character scales (vertical / horizontal).
Style	Set the character style (bold / underlining).
X Position	Set the horizontal start position.
Feed Unit	Set the paper feed amount.

3 Tap (Print) to print.

Graphic printing

Execute the graphic printing according to the following procedure:

1 Tap (Select Image) to select an image file to print.

2 Tap (Color Mode) to select the tone.

> [Gray 16] (multiple tone printing) can only be selected on the TM-T88V model.

3 Tap (Halftone Method) to select the halftone treatment method.

4 Tap (Brightness) and input a value to specify brightness.

5 Tap (Print) to print.

Barcode printing

Execute the barcode printing according to the following procedure:

1 Set the following for barcodes:

Setting	Description
Type	Select the barcode type.
Data	Enter the barcode data.
HRI	Set the HRI position.
Font	Set the HRI font.
Module Size(Width, Height)	Set the barcode module size (width / height).

2 Tap (Print) to print.

2D-code printing

Execute the 2D-code printing according to the following procedure:

1 Select the 2D-code type using (Type).

2 Enter the 2D-code data for (Data).

3 Set the following for each 2D-code:

Setting	Description
Error Correction Level (PDF417,QR Code, Aztec Code, DataMatrix)	Set the error correction level.
Module Size(Width, Height)	Set the 2D-code module size (width / height)
Max Size	Set the maximum 2D-code size.

4 Tap (Print) to print.

Printing in page mode

Execute the printing in page mode according to the following procedure:

1 Enter a string to print for (Print Characters).

2 Set the print area using (Print Area).

Setting	Description
X	Set the origin of horizontal axis.
Y	Set the origin of vertical axis.
Width	Set the width for the print area.
Height	Set the height for the print area.

3 Tap (Print) to print.

Paper cutting

Execute the paper cutting according to the following procedure:

1 Set whether to cut after feeding paper using (Type).

2 Tap (Print) and execute cutting operation.

Log output setting

Use the following procedures:

1 Set whether to enable the log output function and the log output destination in (Enabled).

2 Set the following items according to the log output destination.

Setting	Description
IP Address	Specify the IP address for TCP communication.
Port	Specify the port number for TCP communication.
Log Size	Specify the maximum size of log data that can be saved on the device's storage.
Log Level	Set the level of log data to be output.

3 Set the method of saving the settings in (Save Settings Permanently).

4 Tap (Setting) to enable the log output settings.

Execution result

Process execution result

Any of the following will be displayed:

- Result: Any of the following statuses will be displayed:

String displayed	Description
SUCCESS	Succeeded
ERR_PARAM	An invalid parameter was passed.
ERR_ILLEGAL	Used in an illegal manner.
ERR_PROCESSING	Failed to execute the process.
ERR_TIMEOUT	The process was timed out.
ERR_CONNECT	Failed to connect to the device.
ERR_MEMORY	Could not secure the memory required for the process.
ERR_OFF_LINE	Offline.
ERR_FAILURE	Another error occurred.

- Status: Any of the following printer statuses will be displayed:

String displayed	Description
NO_RESPONSE	No response from the printer
PRINT_SUCCESS	Printing is successfully completed
DRAWER_KICK	Status of the 3rd pin of the drawer kick-out connector = "H" (Other than TM-P60, TM-P60II)
BATTERY_OFFLINE	Battery offline (TM-P60, TM-P60II)
OFF_LINE	Offline
COVER_OPEN	The cover is open
PAPER_FEED	Paper is being fed by a paper feed switch operation
WAIT_ON_LINE	Waiting to be brought back online
PANEL_SWITCH	The paper feed switch is being pressed (ON)
MECHANICAL_ERR	A mechanical error occurred
AUTOCUTTER_ERR	An autocutter error occurred
UNRECOVER_ERR	An unrecoverable error occurred
AUTORECOVER_ERR	An automatically recoverable error occurred
RECEIPT_NEAR_END	No paper in roll paper near end sensor
RECEIPT_END	No paper in roll paper end sensor
BUZZER	Buzzer is sounding (compatible devices only)

- Battery Status: The following will be displayed.

String displayed	Description
0xnnnn	Battery status value For details, refer to Battery Status (p.44) .

Method (API) execution error

Any of the following will be displayed:

- Error Code: Any of the following statuses will be displayed:

String displayed	Description
ERR_PARAM	An invalid parameter was passed.
ERR_OPEN	The open process failed.
ERR_CONNECT	Failed to connect to the device.
ERR_TIMEOUT	All data couldn't be sent during the specified time.
ERR_MEMORY	Could not secure the memory required for the process.
ERR_ILLEGAL	Used in an illegal manner.
ERR_PROCESSING	Failed to execute the process.
ERR_UNSUPPORTED	An unsupported model or language of use has been specified.
ERR_OFF_LINE	Printer is offline.
ERR_FAILURE	Another error occurred.

- Method: The API in which a method execution error occurred is displayed.

Acquisition of Printer Model Name

A command transmission/reception API is used for acquisition of printer model name. For details, refer to Command Transmission/Reception (p.155).

Use the following procedure:

1 Start the sample program. For details, refer to Environmental Construction (p.17).

2 Search for printers. Tap (Printer Discovery) on the main screen.
When (Device Type) is selected, the printers detected by the search are displayed in list form.

3 Select the printer to use among the printers displayed in list form in procedure 2.

4 Tap (Get Printer Name) on the main screen.

5 Tap (Get Printer Name).

6 The following will be displayed.

Content displayed	Description
Printer Name	Displays the model name of the printer.
Language	Displays the language specifications of the printer.

Programming Guide

This chapter describes how to write programs in the application development using ePOS-Print SDK.

> For ways to construct a development environment for Android applications that use ePOS-Print SDK for Android, please refer to the "ePOS-Print SDK for Android Application Development - Setup Guide".

How to Incorporate the ePOS-Print SDK for Android

This section explains how to incorporate the ePOS-Print SDK for Android.

> This explanation uses Eclipse. If you are using another development environment, please make the appropriate changes.

Incorporate the SDK using following procedures.

1 Create a new project in Eclipse.

2 Copy provided JAR file (ePOS-Print.jar) into following path:

```
└libs
    └ePOS-Print.jar
```

3 In Libraries tab of the target project's properties, confirm that the JAR file you added (ePOS-Print.jar) is registered in (Java Build Path).
If it has not been added, add the JAR file into build path using (Add Jars...).

4 Copy the library file (libeposprint.so) into following path:

```
└libs
    └armeabi
        └libeposprint.so
```

5 Select the project in Eclipse's Package Explorer, right click on it, and press (Refresh).

6 Write the package import declaration in the *.java source file(s) of the application you would like to use this SDK in as follows:

```
import com.epson.eposprint.*;

import com.epson.epsonio.*;
```

7 Confirm that the target project's "/libs" folder is in the Source tab of the target project's properties. If not, add "libs" to the build path using (Add Folder...).

8 With the target project selected from Eclipse's Package Explorer, select (Preferences) in the (Window) menu.

9 The (Preferences) screen is displayed. From the list on the left, select (Java)-(Compiler).

10 The (Compiler) screen is displayed. Set the (Compiler compliance level:) to "1.6", and click (Apply). After that, click (OK).

11 Double-click (AndroidManifest.xml) from Eclipse's Package Explorer.

12 Select the (Permissions) tab.

13 The (Android Manifest Permissions) screen is displayed. Click the (Add) button.

14 Select (Uses Permission), and click the (OK) button.

15 (Uses Permission) is added to (Permissions). Select the permissions of functionalities attached to the added (Uses Permission) from the (Name) under (Attributes for Uses Permission).

Functionality	(Name) setting
Wi-Fi	android.permission.INTERNET
Bluetooth	android.permission.BLUETOOTH
	android.permission.BLUETOOTH_ADMIN

> There is one setting of permissions for functionalities that can be attached per [Uses Permission] in [Permissions]. For using the Bluetooth functionality and all functionalities, you must repeat settings from procedures 13 to 15.

16 Save "AndroidManifest.xml".

ePOS-Print SDK

Print Mode

There are two types of print modes: standard and page modes.

Standard mode

In standard mode, characters are printed line by line. The line feed space is adjusted based on the font size and the height of images, barcodes, etc. This mode is suitable for the type of printing such as printing receipts that requires the paper length to change according to the print space.

Page mode

In page mode, you set a print area, lay out data in it, and print the data in a batch operation. Characters, images, and barcodes are laid out in the print positions (coordinates).

Programming Flow

Perform programming following this flow.

1. Print Document Creation (p.31) *
   ❏ Starting the printer search (p.30)
   ❏ Getting the printer search result. (p.30)
   ❏ Stopping the printer search (p.30)

2. Print Document Creation (p.31)
   ❏ To create a text print document: (p.31)
   ❏ To create a graphic print document: (p.32)
   ❏ To create a page mode print document (p.33)

3. Transmission of Print Document (p.34)

*This is optional.

> To ensure successful print operation, write a program in such a way that data is sent after checking the printer status. For the above procedure, refer to Printing After Checking the Printer Status (p.35).

Printer search

Starting the printer search

Use the Finder class's start (p.148) to start searching for printers. Please refer to the following code.

```
int errStatus = IoStatus.SUCCESS;

//Start search
try {
    Finder.start(getBaseContext(), DevType.TCP, "255.255.255.255");
//Exception handling
} catch ( EpsonIoException e ) {
    errStatus = e.getStatus();
}
```

Getting the printer search result.

Use the Finder class's getResult (p.150) to get the result of the printer search.
 Please refer to the following code.

```
int errStatus = IoStatus.SUCCESS;
String[] mList = null;

//Get device list
try {
    mList = Finder.getResult();
//Exception handling
} catch ( EpsonIoException e ) {
    errStatus = e.getStatus();
}
```

Since the printer search takes time to complete, you might not receive any search results if you call the Finder class's getResult immediately after you call start.

Stopping the printer search

Use the Finder class's stop (p.149) to stop searching for printers. Please refer to the following code.

```
int errStatus = IoStatus.SUCCESS;

//Stop search
try {
    Finder.stop();
//Exception handling
} catch ( EpsonIoException e ) {
    errStatus = e.getStatus();
}
```

Print Document Creation

Create a print document using the Builder class (p.45).
Create a Builder class using the constructor for it and create a print document using APIs of the Builder class. Use the programming example below for your reference.

```
try {
    //Initialize a Builder class instance
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    //Create a print document
    builder.addTextLang(Builder.LANG_EN);
    builder.addTextSmooth(Builder.TRUE);
    builder.addTextFont(Builder.FONT_A);
    builder.addTextSize(3, 3);
    builder.addText("Hello,\t");
    builder.addText("World!\n");
    builder.addCut(Builder.CUT_FEED);
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

To create a text print document:

To create a text print document, using APIs for text, store the font settings in command buffers to create a print document. Use the programming example below for your reference.

For the string "Hello, World!", to create a print document based on the following settings:

- Font: FontA
- Scale: x 4 (horizontal) and x 4 (vertical)
- Style: Bold

```
try {
    //Initialize a Builder class instance
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);

    //Create a print document
    //<Configure the print character settings>
    builder.addTextLang(Builder.LANG_EN);
    builder.addTextSmooth(Builder.TRUE);
    builder.addTextFont(Builder.FONT_A);
    builder.addTextSize(4, 4);
    builder.addTextStyle(Builder.FALSE, Builder.FALSE, Builder.TRUE,
                         Builder.PARAM_UNSPECIFIED);

    //<Specify the print data>
    builder.addText("Hello,\t");
    builder.addText("World!\n");
    builder.addCut(Builder.CUT_FEED);
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

3

To create a graphic print document:

To create a graphic print document, for graphics, store the android.graphics.Bitmap class in the command buffers with addImage (p.70) of the Builder class.

Use the programming example below for your reference.

```
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
try {
    //Initialize a Builder class instance
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);

    //Create a print document
    Bitmap bmp = BitmapFactory.decodeResource(getResources(),R.drawable.background);
    builder.addImage(bmp, 0, 0, 8, 48, Builder.PARAM_DEFAULT);
    builder.addCut(Builder.CUT_FEED);
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

For ways of graphic printing, you can also print the graphics registered in the printer's NV memory. For details, refer to addLogo (p.72).

To create a page mode print document

The page mode starts by storing addPageBegin (p.83) of the Builder class into a command buffer. Store the print area (addPageArea (p.85)) and the print start position (addPagePosition (p.89)) in command buffers. Specify the print start position according to the print data. Then, store APIs in command buffers and create print data. For the page mode end, store addPageEnd (p.84) in a command buffer. Use the programming example below for your reference.

For the string "Hello, World!", to create a print document based on the following settings:

- Page mode print area (in dots):
  Origin of horizontal axis: 100, origin of vertical axis: 50, width: 200, height: 100
- Page mode print positions (in dots):
  Horizontal print position: 0, vertical print position: 42
- Font: FontA
- Scale: x 2 (horizontal) and x 2 (vertical)
- Style: Bold

```
try {
    //Initialize a Builder class instance
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);

    //Create a print document
    //<The page mode starts>
    builder.addPageBegin();
    builder.addPageArea(100, 50, 200, 100);
    builder.addPagePosition(0, 42);
    //<Configure the print character settings>
    builder.addTextLang(Builder.LANG_EN);
    builder.addTextSmooth(Builder.TRUE);
    builder.addTextFont(Builder.FONT_A);
    builder.addTextSize(4, 4);
    builder.addTextStyle(Builder.FALSE, Builder.FALSE, Builder.TRUE,
                         Builder.PARAM_UNSPECIFIED);
    //<Specify the print data>
    builder.addText("Hello,\t");
    builder.addText("World!\n");
    //<The page mode ends>
    builder.addPageEnd();
    builder.addCut(Builder.CUT_FEED);
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

3

Transmission of Print Document

Send a print document using the Print class (p.47). Create a Print class using the constructor for it, use sendData to specify the Builder class instance that stores the command buffers for the print document, and send the document. Use the programming example below for your reference.

```
//Initialize a Print class instance
Print printer = new Print();

int[] status = new int[1];
status[0] = 0;

try {
    //Initialize a Builder class instance
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);

    //Create a print document
    //<The page mode starts>
    builder.addTextLang(Builder.LANG_EN);
    builder.addTextSmooth(Builder.TRUE);
    builder.addTextFont(Builder.FONT_A);
    builder.addTextSize(4, 4);
    builder.addTextStyle(Builder.FALSE, Builder.FALSE, Builder.TRUE, Builder.PARAM_UNSPECIFIED);

    //<Specify the print data>
    builder.addText("Hello,\t");
    builder.addText("World!\n");
    builder.addCut(Builder.CUT_FEED);

    //Send a print document
    //<Start communication with the printer>
    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");
    //<Send data>
    printer.sendData(builder, 10000, status);
    //<End communication with the printer>
    printer.closePrinter();

} catch (EposException e) {
    int errStatus = e.getErrorStatus();
    status[0] = e.getPrinterStatus();
    printer.closePrinter();
}
```

Printing After Checking the Printer Status

To ensure successful print operation, print after checking the printer status.
Send the empty print data, and if the printer is online, print it.

Use the programming example below for your reference.

```
//<Send data for confirmation>
Print printer = new Print();

int[] status = new int[1];
status[0] = 0;

try {
    //Create a print document
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);        (1)
    builder.addText("Hello, World!\n");
    builder.addCut(Builder.CUT_FEED);

    //Initialize a Builder class instance for confirmation
    Builder conBuilder = new Builder("TM-T88V", Builder.MODEL_ANK);     (2)

    //<Start communication with the printer>
    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");

    //<Send data for confirmation>
    printer.sendData(conBuilder, 10000, status);                        (3)

    if((status[0] & Print.ST_OFF_LINE) != Print.ST_OFF_LINE){
                                                                        (4)
        //<Send print data>
        printer.sendData(builder, 10000, status);
    }

    //<End communication with the printer>
    printer.closePrinter();

} catch (EposException e) {
    int errStatus = e.getErrorStatus();
    status[0] = e.getPrinterStatus();
    printer.closePrinter();
}
```

1 Create print data.

2 To check the printer status, send empty print data.

3 Send the print data created in (2).

4 If the print data created in (2) was properly sent, and the printer is online, then send the print data created in (1).

Automatic Acquisition of Printer Status

In ePOS-Print SDK, listener is used to automatically notify printer status to the application.
Refer to the following.

```
//Registration of StatusChangeEventListener for giving notification of printer status        (1)
public class SampleActivity extends Activity implements OnClickListener,
                                              StatusChangeEventListener {

    ///Process///

        //Implement the StatusChangeEventListener method
        private void onStatusChangeEvent(String deviceName, int status) {        (2)/(5)

            ///Process///
        }

    private void openPrinter() {
            //Initialize the print class instance
            Print printer = new Print();

            //Register the notification destination of printer status changes        (3)
            printer.setStatusChangeEventCallback(this);

            try {
            //Start communications with the printer and monitoring of the printer status        (4)
                printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                            Print.PARAM_DEFAULT);

                ///Process///

            } catch (EposException e) {
                int errStatus = e.getErrorStatus();
                printer.closePrinter();
            }
        }
}
```

1   Define the listener interface for acquiring the printer status.

> Above, StatusChangeEventListener, which notifies printer status at the intervals
> specified in openPrinter(For acquiring printer status) (p.107), is defined.
> ePOS-Print has listener interfaces according to each printer status, for example, events
> such as cover open and drawer open. Use these according to the desired purpose of
> use. See the Listener Interface List (p.38) for the listener interface that can be used with
> ePOS-Print.

2   Implement the notification destination method when events occur.

3   Register the printer status notification destination.

4   Use openPrinter(For acquiring printer status) (p.107) to start monitoring of the printer status.

5   Notify the printer status to the event implemented in (2).

> When printer status notification is ended, it ends on the closePrinter (p.111) of the Print class.

Listener Interface List

> For details on the listener interface, refer to API Reference (p.45), which explains the notification destination registration API.

Function	Event listener
	Notification destination method
	Notification destination registration API
Printer status notification	public interface StatusChangeEventListener extends EventListener
	void onStatusChangeEvent(String deviceName, int status)
	setStatusChangeEventCallback (p.116)
Online notification	public interface OnlineEventListener extends EventListener
	void onOnlineEvent(String deviceName)
	setOnlineEventCallback (p.118)
Offline notification	public interface OfflineEventListener extends EventListener
	void onOfflineEvent(String deviceName)
	setOfflineEventCallback (p.120)
Power off notification	public interface PowerOffEventListener extends EventListener
	void onPowerOffEvent(String deviceName)
	setPowerOffEventCallback (p.122)
Cover close notification	public interface CoverOkEventListener extends EventListener
	void onCoverOkEvent(String deviceName)
	setCoverOkEventCallback (p.124)
Cover open notification	public interface CoverOpenEventListener extends EventListener
	void onCoverOpenEvent(String deviceName)
	setCoverOpenEventCallback (p.126)
Paper OK notification	public interface PaperOkEventListener extends EventListener
	void onPaperOkEvent(String deviceName)
	setPaperOkEventCallback (p.128)
Paper near end notification	public interface PaperNearEndEventListener extends EventListener
	void onPaperNearEndEvent(String deviceName)
	setPaperNearEndEventCallback (p.130)
Paper end notification	public interface PaperEndEventListener extends EventListener
	void onPaperEndEvent(String deviceName)
	setPaperEndEventCallback (p.132)
Drawer close notification	public interface DrawerClosedEventListener extends EventListener
	void onDrawerClosedEvent(String deviceName)
	setDrawerClosedEventCallback (p.134)

Function	Event listener
	Notification destination method
	Notification destination registration API
Drawer open notification	public interface DrawerOpenEventListener extends EventListener
	void onDrawerOpenEvent(String deviceName)
	setDrawerOpenEventCallback (p.136)
Battery low notification	public interface BatteryLowEventListener extends EventListener
	void onBatteryLowEvent(String deviceName)
	setBatteryLowEventCallback (p.138)
Battery OK notification	public interface BatteryOkEventListener extends EventListener
	void onBatteryOkEvent(String deviceName)
	setBatteryOkEventCallback (p.140)
Battery status notification	public interface BatteryStatusChangeEventListener extends EventListener
	void onBatteryStatusChangeEvent(String deviceName, int battery)
	setBatteryStatusChangeEventCallback (p.142)

3

Exception handling

In ePOS-Print SDK for Android, it is designed that when an error occurs, a propriety exception with an integer (int) type parameter is generated to notify the calling side of such an error. The ePOS-Print API acquires information using the EposException class (p.47), and the search API acquires information using the EposException class (p.47). The following errors are sent:

Type	Description
Error status	Cause of error when each class's API was executed. For details, refer to Error Status List (p.42).
Printer status	Status of the printer when print data was sent. The printer status can be acquired only when sendData (p.112) is executed. For details, refer to Printer Status List (p.43).
Battery status	Status of the printer's battery. For details, refer to Battery Status (p.44).

Steps for Handling

ePOS-Print API

Acquire the error status using the EposException class getErrorStatus (p.144), the printer status using getPrinterStatus (p.145), and the battery status using getBatteryStatus (p.146).

Use the programming example below for your reference.

```
//<Send data for confirmation>
Print printer = new Print();

int[] status = new int[1];
int[] battery = new int[1]
status[0] = 0;
battery[0] = 0;

try {
    //Create a print document
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addText("Hello,\t");
    builder.addText("World!\n");
    builder.addCut(Builder.CUT_FEED);

    //<Send print data>
    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");
    printer.sendData(builder, 10000, status, battery);
    printer.closePrinter();

} catch (EposException e) {
    //Acquire the error status
    int errStatus = e.getErrorStatus();
    //Acquire the printer status
    status[0] = e.getPrinterStatus();
    //Acquire the battery status
    battery[0] = e.getBatteryStatus();
    printer.closePrinter();
}
```

Search API

Acquire the error status using getStatus (p.151) of the EpsonIoException class.

Use the programming example below for your reference.

```
int errStatus = IoStatus.SUCCESS;
String[] mList = null;

//Acquire a list of devices
try {
    Finder.start(getBaseContext(), DevType.TCP, "255.255.255.255");
    mList = Finder.getResult();
//Exception processing
} catch ( EpsonIoException e ) {
    errStatus = e.getStatus();
}
```

3

Error Status List

Error statuses are defined in each API-executing class.

Error status	Cause
ERR_PARAM	Invalid parameter was passed. <Example> • An invalid parameter such as null was passed. • A value outside the supported range was specified.
ERR_OPEN	Open processing failed. <Example> Could not connect to the designated printer.
ERR_CONNECT	Failed to connect to device. <Example> Failed to send the data to the printer.
ERR_TIMEOUT	The specified timeout time was exceeded. <Example> Could not transmit all the data in the specified time.
ERR_MEMORY	Could not allocate the necessary memory for processing.
ERR_ILLEGAL	Illegal method used. <Example> When the printer was not opened, an API for sending a command to the printer was called.
ERR_PROCESSING	Could not execute process. <Example> Could not execute the process because an identical process is being executed in another thread.
ERR_UNSUPPORTED	An unsupported model name or language specification was specified.
ERR_OFF_LINE	The printer is offline.
ERR_FAILURE	An unspecified error occurred.

Printer Status List

Printer Status	Cause
Print.ST_NO_RESPONSE (0x00000001)	No response from the printer
Print.ST_PRINT_SUCCESS (0x00000002)	Printing is successfully completed
<Other than TM-P60, TM-P60II> Print.ST_DRAWER_KICK (0x00000004)	Status of the 3rd pin of the drawer kick-out connector = "H"
<TM-P60, TM-P60II> Print.ST_BATTERY_OFFLINE (0x00000004)	Battery offline status
Print.ST_DRAWER_KICK (0x00000004)	Status of the 3rd pin of the drawer kick-out connector = "H"
Print.ST_OFF_LINE 0x00000008)	Offline
Print.ST_COVER_OPEN (0x00000020)	The cover is open
Print.ST_PAPER_FEED (0x00000040)	Paper is being fed by a paper feed switch operation
Print.ST_WAIT_ON_LINE (0x00000100)	Waiting to be brought back online
Print.ST_PANEL_SWITCH (0x00000200)	The paper feed switch is being pressed (ON)
Print.ST_MECHANICAL_ERR (0x00000400)	A mechanical error occurred
Print.ST_AUTOCUTTER_ERR (0x00000800)	An autocutter error occurred
Print.ST_UNRECOVER_ERR (0x00002000)	An unrecoverable error occurred
Print.ST_AUTORECOVER_ERR (0x00004000)	An automatically recoverable error occurred
Print.ST_RECEIPT_NEAR_END (0x00020000)	No paper in roll paper near end sensor
Print.ST_RECEIPT_END (0x00080000)	No paper in roll paper end sensor
Print.ST_BUZZER (0x01000000)	• A buzzer is on (only for applicable devices) • Waiting for label to be removed (only for applicable devices)

3

Battery Status

The battery status consists of the following 16 bits (0x0000).

Bit	Description
Upper 8 bits	Common battery status For details, refer to Common battery status (upper 8 bits) (p.44) .
Lower 8 bits	Battery status exclusive by model For details, refer to Printer specifications (p.165).

"0x0000" is returned if the battery status cannot be acquired or if the model does not support the battery status.

Common battery status (upper 8 bits)

Battery Status	Cause
0x30	The AC adapter is connected
0x31	The AC adapter is not connected

API Reference

This chapter describes the APIs provided in the ePOS-Print SDK for Android.

ePOS-Print API

The ePOS-Print APIs are APIs for creating and printing print documents. The following classes are available.

❏   Builder class (p. 45)

❏   Print class (p. 47)

❏   EposException class (p. 47)

> The APIs that you can use and the settings that you can designate vary based on the printer. For details, refer to Printer specifications (p.165).

Builder class

This class creates print documents for printer control commands such as character strings to print, graphic printing, and paper cutting. The following APIs are available.

API		Description	Page
Constructor		Initialize a Builder class instance.	48
Constructor(For log output)		Initialize a Builder class instance. This constructor is for the log output function.	50
Clearing command buffers	clearCommandBuffer	Clears the command buffers added by APIs.	52
Text	addTextAlign	Adds a tag for the text alignment setting.	53
	addTextLineSpace	Adds a tag for the line feed space setting.	54
	addTextRotate	Adds a tag for the text rotation setting.	55
	addText	Adds a tag for printing text.	56
	addTextLang	Adds a tag for the target language setting.	57
	addTextFont	Adds a tag for the text font setting.	58
	addTextSmooth	Adds a tag for the text smoothing setting.	59
	addTextDouble	Adds a tag for specifying the double-sized text setting.	60
	addTextSize	Adds a tag for the text scale setting.	61
	addTextStyle	Adds a tag for the text style setting.	62
	addTextPosition	Adds a tag for specifying the print position of text.	64
Paper Feed	addFeedUnit	Adds a tag for paper feeding (in dots).	65
	addFeedLine	Adds a tag for paper feeding (in lines).	66
	addPaperFeedPosition	Adds a tag for label / black mark paper feeding.	101

4

API		Description	Page
Graphic	addImage (For multiple tone printing)	Adds multiple tone raster image printing to the command buffer.	67
	addImage	Adds a tag for a raster image to be printed.	70
	addLogo	Adds a tag for an NV logo to be printed.	72
Barcode	addBarcode	Adds a tag for a bar code to be printed.	73
	addSymbol	Adds a tag for a 2D-Code to be printed.	78
Pagemode	addPageBegin	Adds a tag for switching to page mode.	83
	addPageEnd	Adds a tag for finishing page mode.	84
	addPageArea	Adds a tag for specifying the print area in page mode.	85
	addPageDirection	Adds a tag for specifying the print direction in page mode.	87
	addPagePosition	Adds a tag for specifying the print position in page mode.	89
	addPageLine	Adds a tag for drawing a line in page mode.	91
	addPageRectangle	Adds a tag for drawing a rectangle in page mode.	93
Cut	addCut	Adds a tag for paper cut.	95
Drawer kick-out	addPulse	Adds a tag for the drawer kick-out.	96
Buzzer	addSound (For setting cycle buzzer)	Sets the buzzer sounding cycle and adds it to the command buffer.	97
	addSound	Adds a tag for turning on the buzzer.	99
Paper Layout	addLayout	Adds a tag for paper layout information.	102
Send Command	addCommand	Adds a tag for inserting commands.	104

Print class

Controls the printer by sending a print document created using the Builder class, and monitors the transmission result and the communication status.

API	Description	Page
Constructor	Initialize a Print class instance.	105
Constructor(For log output)	Initialize a Print class instance. This constructor is for the log output function.	106
openPrinter (For acquiring printer status)	Starts communications with the printer and monitoring of the printer status	107
openPrinter	Start communication with the printer.	109
closePrinter	End communication with the printer.	111
sendData	Sends a command to the printer.	112
sendData (For acquiring battery status)	Sends a command to the printer.	114
setStatusChangeEventCallback	Registers the printer status notification destination	116
setOnlineEventCallback	Registers the online event notification destination	118
setOfflineEventCallback	Registers the offline event notification destination	120
setPowerOffEventCallback	Registers the power off event notification destination	122
setCoverOkEventCallback	Registers the cover close event notification destination	124
setCoverOpenEventCallback	Registers the cover open event notification destination	126
setPaperOkEventCallback	Registers the paper OK event notification destination	128
setPaperNearEndEventCallback	Registers the paper near end event notification destination	130
setPaperEndEventCallback	Registers the paper end event notification destination	132
setDrawerClosedEventCallback	Registers the drawer close event notification destination	134
setDrawerOpenEventCallback	Registers the drawer open event notification destination	136
setBatteryLowEventCallback	Registers the battery low event notification destination	138
setBatteryOkEventCallback	Registers the battery OK event notification destination	140
setBatteryStatusChangeEvent- Callback	Registers the battery status notification destination	142

EposException class

Acquires the status during slowdown when an exception occurs in printing error or an API execution error occurs.

API	Description	Page
getErrorStatus	Acquires the error status.	144
getPrinterStatus	Acquires the printer status.	145
getBatteryStatus	Acquires the battery status.	146

Builder class (Constructor)

Constructor for the Builder class. Initializes a Builder class instance.

> To use the log output function, use the

Syntax

```
public Builder(String printerModel, int lang)
 throws EposException
```

Parameter

- printerModel : Specifies the model name for the target printer.

Set value	Description
"TM-T88V"	TM-T88V
"TM-T70"	TM-T70
"TM-T70II"	TM-T70II
"TM-U220"	TM-U220
"TM-P60"	TM-P60
"TM-P60II"	TM-P60II
"TM-T20"	TM-T20
"TM-T81II"	TM-T81II
"TM-T82"	TM-T82
"TM-T82II"	TM-T82II
"TM-T90II"	TM-T90II

- lang : Specifies the language specifications for the printer.

Set value	Description
Builder.MODEL_ANK	ANK model
Builder.MODEL_JAPANESE	Japanese model
Builder.MODEL_CHINESE	Chinese model
Builder.MODEL_TAIWAN	Taiwanese model
Builder.MODEL_KOREAN	Korean model
Builder.MODEL_THAI	Thai model
Builder.MODEL_SOUTHASIA	South Asian model

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_UNSUPPORTED	An unsupported model name or unsupported language specifications were specified.
ERR_FAILURE	An unspecified error occurred.

Example

If you are initializing the command buffer for the TM-T88V ANK model:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

4

Builder class (Constructor) (For log output)

Constructor for the Builder class. Initializes a Builder class instance.
Use this constructor to use the log output function.

Syntax

```
public Builder(String printerModel, int lang,
               Context context) throws EposException
```

Parameter

- printerModel : Specifies the model name for the target printer.

Set value	Description
"TM-T88V"	TM-T88V
"TM-T70"	TM-T70
"TM-T70II"	TM-T70II
"TM-U220"	TM-U220
"TM-P60"	TM-P60
"TM-P60II"	TM-P60II
"TM-T20"	TM-T20
"TM-T81II"	TM-T81II
"TM-T82"	TM-T82
"TM-T82II"	TM-T82II
"TM-T90II"	TM-T90II

- lang : Specifies the language specifications for the printer.

Set value	Description
Builder.MODEL_ANK	ANK model
Builder.MODEL_JAPANESE	Japanese model
Builder.MODEL_CHINESE	Chinese model
Builder.MODEL_TAIWAN	Taiwanese model
Builder.MODEL_KOREAN	Korean model
Builder.MODEL_THAI	Thai model
Builder.MODEL_SOUTHASIA	South Asian model

- context : Specifies the context of the application.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_UNSUPPORTED	An unsupported model name or unsupported language specifications were specified.
ERR_FAILURE	An unspecified error occurred.

Example

If you are initializing the command buffer for the TM-T88V ANK model:

```
import android.content.Context;

try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK,
                                  getApplicationContext());
   ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

4

clearCommandBuffer

Clears command buffers used by APIs of the Builder class.

Syntax

```
public void clearCommandBuffer()
```

Example

If you are clearing the command buffer:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    ///Process///
    builder.clearCommandBuffer();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addTextAlign

Adds the text alignment setting to the command buffer.

	This API setting also applies to barcodes/2D-Code.
⚠	When the page mode is selected for the print mode, use addPagePosition (p.89) instead of this API to set the alignment.

Syntax

```
public void addTextAlign(int align) throws EposException
```

Parameter

- align : Specifies the text alignment.

Set value	Description
Builder.ALIGN_LEFT (default)	Alignment to the left
Builder.ALIGN_CENTER	Alignment to the center
Builder.ALIGN_RIGHT	Alignment to the right

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set alignment to the center:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextAlign(Builder.ALIGN_CENTER);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

53

addTextLineSpace

Adds the line feed space setting to the command buffer.

Syntax

```
public void addTextLineSpace(int linespc)
  throws EposException
```

Parameter

- linespc :   Specifies the line feed space (in dots). Specifies an integer from 0 to 255.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set the line feed space to 30 dots:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextLineSpace(30);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addTextRotate

Adds the text rotation setting to the command buffer.

> This API setting also applies to barcodes/two dimensional symbols.

> When the page mode is selected for the print mode, to set text rotation, use the addPageDirection (p.87) instead of this API function.

Syntax

```
public void addTextRotate(int rotate)
  throws EposException
```

Parameter

- rotate :   Specifies whether to rotate text.

Set value	Description
Builder.TRUE	Specifies rotated printing of text.
Builder.FALSE (default)	Cancels rotated printing of text.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set text rotation:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextRotate(Builder.TRUE);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addText

Adds the printing of text to the command buffer.

> ⚠ After printing text, to print content other than text, execute line feed or paper feed.

Syntax

```
public void addText(String data) throws EposException
```

Parameter

- data :   Specify a character string to be printed.
            For the horizontal tab/line feed, use the following escape sequences:

String	Description
\t	Horizontal tab(HT)
\n	Line feed (LF)
\\	Carriage return

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To add character strings:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addText("Hello,\t");
    builder.addText("World!\n");
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addTextLang

Adds the language setting to a command buffer. Encodes the string specified by addText (p.56) according to the language information specified by this API.

This API is an API to be called before calling addText (p.56).

Syntax

```
public void addTextLang(int lang) throws EposException
```

Parameter

- lang :   Specifies the target language.

Set value	Language
Builder.LANG_EN(default)	English(ANK)
Builder.LANG_JA	Japanese
Builder.LANG_ZH_CN	Simple Chinese
Builder.LANG_ZH_TW	Traditional Chinese
Builder.LANG_KO	Korean
Builder.LANG_TH	Thai (South Asia specifications)
Builder.LANG_VI	Vietnamese (South Asia specifications)

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set the language as English:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextLang(Builder.LANG_EN);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

4

addTextFont

Adds the text font setting to the command buffer.

Syntax

```
public void addTextFont(int font) throws EposException
```

Parameter

- font :   Specifies the font.

Set value	Language
Builder.FONT_A (default)	Font A
Builder.FONT_B	Font B
Builder.FONT_C	Font C

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set the font B:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextFont(Builder.FONT_B);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addTextSmooth

Adds the smoothing setting to the command buffer.

Syntax

```
public void addTextSmooth(int smooth)
  throws EposException
```

Parameter

- smooth :   Specifies whether to enable smoothing.

Set value	Description
Builder.TRUE	Specifies smoothing.
Builder.FALSE (default)	Cancels smoothing

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To enable smoothing:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextSmooth(Builder.TRUE);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

4

addTextDouble

Adds the double-sized text setting to the command buffer.

Syntax

```
public void addTextDouble(int dw, int dh)
  throws EposException
```

Parameter

- dw :   Specifies the double-sized width.

Set value	Description
Builder.TRUE	Specifies the double-sized width.
Builder.FALSE (default)	Cancels the double-sized width
Builder.PARAM_UNSPECIFIED	Retains the current setting.

- dh :   Specifies the double-sized height.

Set value	Description
Builder.TRUE	Specifies the double-sized height
Builder.FALSE (default)	Cancels the double-sized height
Builder.PARAM_UNSPECIFIED	Retains the current setting.

> When Builder.true or 1 is set for both the dw and dh parameters, double width and height characters are printed.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set the size as double width and height:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextDouble(Builder.TRUE, Builder.TRUE);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addTextSize

Adds the text scale setting to the command buffer.

Syntax

```
public void addTextSize(int width, int height)
  throws EposException
```

Parameter

- width :   Specifies the horizontal scale of text.

Set value	Description
Integer from 1 to 8	Horizontal scale (default : 1)
Builder.PARAM_UNSPECIFIED	Retains the current setting.

- height :   Specifies the vertical scale of text.

Set value	Description
Integer from 1 to 8	**Vertical scale** (default : 1)
Builder.PARAM_UNSPECIFIED	Retains the current setting.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set a horizontal scale of x 4 and a vertical scale of x 4:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextSize(4, 4);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

4

addTextStyle

Adds the text style setting to the command buffer.

Syntax

```
public void addTextStyle(int reverse, int ul, int em
, int color) throws EposException
```

Parameter

- reverse :   Specifies inversion of black and white for text.

Set value	Description
Builder.TRUE	Specifies the inversion of black and white parts of characters.
Builder.FALSE (default)	Cancels the inversion of black and white parts of characters.
Builder.PARAM_UNSPECIFIED	Retains the current setting.

- ul :   Specifies the underline style.

Set value	Description
Builder.TRUE	Specifies underlining.
Builder.FALSE (default)	Cancels underlining.
Builder.PARAM_UNSPECIFIED	**Retains the current setting.**

- em :   Specifies the bold style.

Set value	Description
Builder.TRUE	Specifies emphasized printing of characters.
Builder.FALSE (default)	Cancels emphasized printing of characters.
Builder.PARAM_UNSPECIFIED	Retains the current setting.

- color :   Specifies the color.

Set value	Description
Builder.COLOR_NONE	Characters are not printed.
Builder.COLOR_1 (default)	First color
Builder.COLOR_2	Second color
Builder.COLOR_3	Third color
Builder.COLOR_4	Fourth color
Builder.PARAM_UNSPECIFIED	Retains the current color setting

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set the underline style:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextStyle(Builder.PARAM_UNSPECIFIED, Builder.TRUE,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

4

63

addTextPosition

Adds the horizontal print start position of text to the command buffer.

Syntax

```
public void addTextPosition(int x) throws EposException
```

Parameter

- x :   Specifies the horizontal print start position (in dots).
                   Specifies an integer from 0 to 65535.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set the print position at 120 dots from the left end:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextPosition(120);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addFeedUnit

Adds paper feeding in dots to the command buffer.

Syntax

```
public void addFeedUnit(int unit) throws EposException
```

Parameter

- unit : Specifies the paper feed space (in dots). Specifies an integer from 0 to 255.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To feed paper by 30 dots:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addFeedUnit(30);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

4

addFeedLine

Adds paper feeding in lines to the command buffer.

Syntax

```
public void addFeedLine(int line) throws EposException
```

Parameter

- unit :   Specifies the paper feed space (in lines). Specifies an integer from 0 to 255.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To feed paper by 3 lines:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addFeedLine(3);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addImage (For multiple tone printing)

Adds raster image printing to the command buffer.

Prints the graphic in the android.graphics.Bitmap class.

Out of the android.graphics.Bitmap class graphics, the specified scope is converted to raster image data according to this API setting. 1 pixel of the image corresponds to 1 dot of the printer. If transparent shading is included, it is regarded as white.

> • To print a raster image at high speed, specify Builder.ALIGN_LEFT for the addTextAlign (p.53), and specify a multiple of 8 not exceeding the printer's paper width for the width parameter of this API.
> • When printing transmission images, the printing speed may become slower.

Syntax

```
public void addImage(Bitmap data, int x, int y
                  , int width, int height, int color
                  , int mode, int halftone
                  , double brightness)
                   throws EposException
```

Parameter

- data : Specifies an instance of the android.graphics.Bitmap class.
- x : Specifies the horizontal start position in the print area. Specifies an integer from 0 to 65534.
- y : Specifies the vertical start position in the print area. Specifies an integer from 0 to 65534.
- width : Specifies the width of the print area. Specifies an integer from 1 to 65535.
- height : Specifies the height of the print area. Specifies an integer from 1 to 65535.

> If the area specified by the x/y parameters and the width/height parameters extends beyond the image size specified by the data parameter, an EposException with ERR_PARAM contained in its error status occurs.

- color : Specifies the color.

Set value	Description
Builder.COLOR_NONE	Characters are not printed.
Builder.COLOR_1	First color
Builder.COLOR_2	Second color
Builder.COLOR_3	Third color
Builder.COLOR_4	Fourth color
Builder.PARAM_DEFAULT	First color

4

- mode : Specify the color mode.

Set value	Description	TM printer-separate setting										
		TM-T88V	TM-T70	TM-T70II	TM-U220	TM-P60	TM-P60II	TM-T20	TM-T81II	TM-T82	TM-T82II	TM-T90II
Builder.MODE_MONO	Monochrome (2 tone)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Builder.MODE_GRAY16	Multiple tone (16 tone)	✔	-	✔	-	-	-	-	-	-	-	✔
Builder.PARAM_DEFAULT	Specify the half tone treatment method. (Monochrome (2 tone))	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

- halftone : Specify the half tone treatment method.

Set value	Description
Builder.HALFTONE_DITHER	Dither (This is suitable for graphic printing).
Builder.HALFTONE_ERROR_DIFFUSION	Error diffusion (This is suitable for mixed printing or characters and graphics).
Builder.HALFTONE_THRESHOLD	Threshold value (This is suitable for printing of characters).
Builder.PARAM_DEFAULT	Default value (dither) selection

⚠ In the case of multiple tone (16 tone), this is disregarded.

- brightness : Specify the correction value for brightness.

Set value	Description
Actual figure from 0.1 to 10.0	Brightness correction value (gamma value)
Builder.PARAM_DEFAULT	Select the default value (1.0)

⚠ If you specify a value other than 1.0, the printing speed will become slower.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

```
try {
    Bitmap imageData = null;
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    ///Process///
    builder.addImage(imageData, 0, 0, 256, 256, Builder.PARAM_DEFAULT,
                    Builder.MODE_MONO, Builder.HALFTONE_DITHER, 1.0);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

To print an image 256 dots wide and 256 dots high in page mode:

```
try {
    Bitmap imageData = null;
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    ///Process///
    builder.addPageBegin();
    builder.addPagePosition(0, 255);
    builder.addImage(imageData, 0, 0, 256, 256, Builder.PARAM_DEFAULT,
                    Builder.MODE_MONO, Builder.HALFTONE_DITHER, 1.0);
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

4

addImage

Adds raster image printing to the command buffer.

Prints the graphic in the android.graphics.Bitmap class.

Of the graphics in the android.graphics.Bitmap class, makes the specified range into binary with the dither processing, and converts it into raster image data. 1 pixel of the image corresponds to 1 dot of the printer. If transparent shading is included, it is regarded as white.

> • When printing in multiple tone, use addImage (For multiple tone printing) (p.67).
> • To print a raster image at high speed, specify Builder.ALIGN_LEFT for the addTextAlign (p.53), and specify a multiple of 8 not exceeding the printer's paper width for the width parameter of this API.
> • When printing transmission images, the printing speed may become slower.

Syntax

```
public void addImage(Bitmap data, int x, int y
                 , int width, int height, int color)
                    throws EposException
```

Parameter

- data : Specifies an instance of the android.graphics.Bitmap class.
- x : Specifies the horizontal start position in the print area. Specifies an integer from 0 to 65534.
- y : Specifies the vertical start position in the print area. Specifies an integer from 0 to 65534.
- width : Specifies the width of the print area. Specifies an integer from 1 to 65535.
- height : Specifies the height of the print area. Specifies an integer from 1 to 65535.
- color : Specifies the color.

Set value	Description
Builder.COLOR_NONE	Characters are not printed.
Builder.COLOR_1	First color
Builder.COLOR_2	Second color
Builder.COLOR_3	Third color
Builder.COLOR_4	Fourth color
Builder.PARAM_DEFAULT	First color

> If the area specified by the x/y parameters and the width/height parameters extends beyond the image size specified by the data parameter, an EposException with ERR_PARAM contained in its error status occurs.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

```
try {
    Bitmap imageData = null;
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    ///Process///
    builder.addImage(imageData, 0, 0, 256, 256, Builder.PARAM_DEFAULT);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

To print an image 256 dots wide and 256 dots high in page mode:

```
try {
    Bitmap imageData = null;
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    ///Process///
    builder.addPageBegin();
    builder.addPagePosition(0, 255);
    builder.addImage(imageData, 0, 0, 256, 256, Builder.PARAM_DEFAULT);
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

4

addLogo

Adds NV logo printing to the command buffer.

Prints a logo registered in the NV memory of the printer.

> Register a logo in advance into the printer using the following utilities:
> * Model-dedicated Utility
> * TM Flash Logo Setup Utility

Syntax

```
public void addLogo(int key1, int key2)
  throws EposException
```

Parameter

* key1 :   Specifies the key code 1 of an NV logo. Specifies an integer from 0 to 255.
* key2 :   Specifies the key code 2 of an NV logo. Specifies an integer from 0 to 255.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To print a NV logo with the key code parameters specified as 48, 48:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addLogo(48, 48);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addBarcode

Adds barcode printing to the command buffer.

Syntax

```
public void addBarcode
(String data, int type, int hri, int font, int width
, int height) throws EposException
```

Parameter

- data :   Specifies the barcode data as a string.

> Specify a string that follows the barcode standard specified by the type parameter. If the specified
> string does not conform to the standard, a barcode will not be printed.

Barcode type	Description
UPC-A	When an 11-digit number is specified, a check digit is automatically added. When a 12-digit number is specified, the 12th digit is processed as a check digit but the check digit is not validated.
UPC-E	Specify 0 as the first digit. Specify the manufacturer code in the digits 2 to 6. Specify (right-align) the item code in the digits 7 to 11. The number of item code digits varies depending on the manufacturer code. Specify 0s in empty digits.
EAN13	When an 11-digit number is specified, a check digit is automatically added. When a 12-digit number is specified, the 12th digit is processed as a check digit but the check digit is not validated.
JAN13	
EAN8	When a 7-digit number is specified, a check digit is automatically added. When an 8-digit number is specified, the 8th digit is processed as a check digit but the check digit is not validated.
JAN8	
CODE39	When the first character is *, the character is processed as the start character. In other cases, a start character is automatically added.
ITF	Start and stop codes are automatically added. Check digits are not added or validated.
CODABAR	Specify a start character (A to D, a to d). Specify a stop character (A to D, a to d). Check digits are not added or validated.
CODE93	Start and stop characters are automatically added. A check digit is automatically calculated and added.

4

Barcode type	Description
CODE128	Specify a start character (CODE A, CODE B, CODE C). A stop character is automatically added. A check digit is automatically calculated and added. To encode each of the following characters, specify two characters starting with the character "{": FNC1: {1 FNC2: {2 FNC3: {3 FNC4: {4 CODE A: {A CODE B: {B CODE C: {C SHIFT: {S {: {{
GS1-128	A start character, FNC1, a check digit, and a stop character are automatically added. To automatically calculate and add a check digit for an application identifier (AI) and the subsequent data, specify the character "*" in the position of the check digit. You can enclose an application identifier (AI) in parentheses. The parentheses are used as HRI print characters and are not encoded as data. You can insert spaces between an application identifier (AI) and data. The spaces are used as HRI print characters and are not encoded as data. To encode each of the following characters, specify two characters starting with the character "{": FNC1: {1 FNC3: {3 (: {( ): {) *: {* {: {{
GS1 DataBar Omnidi-rectional GS1 DataBar Truncated GS1 DataBar Limited	Specify a 13-digit global trade item number (GTIN) not including an application identifier (AI) or a check digit.

Barcode type	Description
BARCODE_GS1_ DATABAR_EXPANDED	You can enclose an application identifier (AI) in parentheses. The parentheses are used as HRI print characters and are not encoded as data. To encode each of the following characters, specify two characters starting with the character "{": FNC1:　　　　　{1 (:　　　　　　{( ):　　　　　　{)

To specify binary data that cannot be represented by character strings, use the following escape sequences.

String	Description
\xnn	Control code
\\	Back slash

- type :　　　　Specifies the barcode type.

Set value	Barcode type
Builder.BARCODE_UPC_A	UPC-A
Builder.BARCODE_UPC_E	UPC-E
Builder.BARCODE_EAN13	EAN13
Builder.BARCODE_JAN13	JAN13
Builder.BARCODE_EAN8	EAN8
Builder.BARCODE_JAN8	JAN8
Builder.BARCODE_CODE39	CODE39
Builder.BARCODE_ITF	ITF
Builder.BARCODE_CODABAR	CODABAR
Builder.BARCODE_CODE93	CODE93
Builder.BARCODE_CODE128	CODE128
Builder.BARCODE_GS1_128	GS1-128
Builder.BARCODE_GS1_DATABAR_OMNIDIRECTIONAL	GS1 DataBar Omnidirectional
Builder.BARCODE_GS1_DATABAR_TRUNCATED	GS1 DataBar Truncated
Builder.BARCODE_GS1_DATABAR_LIMITED	GS1 DataBar Limited
Builder.BARCODE_GS1_DATABAR_EXPANDED	GS1 Databar Expanded

- hri :　　　　Specifies the HRI position.

Set value	Description
Builder.HRI_NONE(default)	HRI not printed
Builder.HRI_ABOVE	Above the bar code
Builder.HRI_BELOW	Below the bar code
Builder.HRI_BOTH	Both above and below the bar code
Builder.PARAM_UNSPECIFIED	Retains the current setting.

4

- font :   Specifies the HRI font.

Set value	Description
Builder.FONT_A(default)	Font A
Builder.FONT_B	Font B
Builder.FONT_C	Font C
Builder.PARAM_UNSPECIFIED	Retains the current setting.

- width :   Specifies the width of each module in dots. Specifies an integer from 2 to 6.

Set value	Description
Integer from 1 to 6	The width of each module. (Unit:dot)
Builder.PARAM_UNSPECIFIED	Retains the current setting.

- height :   Specifies the barcode height in dots. Specifies an integer from 1 to 255.

Set value	Description
Integer from 1 to 255	The barcode height. (Unit:dot)
Builder.PARAM_UNSPECIFIED	Retains the current setting.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To print barcodes:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addBarcode("01234567890", Builder.BARCODE_UPC_A,
            Builder.HRI_BELOW, Builder.PARAM_UNSPECIFIED, 2, 64);
    builder.addBarcode("01234500005", Builder.BARCODE_UPC_E,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("201234567890", Builder.BARCODE_EAN13,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("201234567890", Builder.BARCODE_JAN13,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("2012345", Builder.BARCODE_EAN8,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("2012345", Builder.BARCODE_JAN8,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("ABCDE", Builder.BARCODE_CODE39,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("012345", Builder.BARCODE_ITF,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("A012345A", Builder.BARCODE_CODABAR,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("ABCDE", Builder.BARCODE_CODE93,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("{Babcde", Builder.BARCODE_CODE128,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("(01)201234567890*", Builder.BARCODE_GS1_128,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("0201234567890",
            Builder.BARCODE_GS1_DATABAR_OMNIDIRECTIONAL,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("0201234567890",
            Builder.BARCODE_GS1_DATABAR_TRUNCATED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("0201234567890",
            Builder.BARCODE_GS1_DATABAR_LIMITED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("(01)2012345678903",
            Builder.BARCODE_GS1_DATABAR_EXPANDED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

4

addSymbol

Adds 2D-Code printing to the command buffer.

Syntax

```
public void addSymbol
(String data, int type, int level, int width,
 int height, int size)
 throws EposException
```

Parameter

- data :   Specifies 2D-Code data as a character string.

2D-Code type	Description
Standard PDF417	Convert the character string to the string in UTF-8, apply the escape sequence, and then encode the string.
Truncated PDF417	The data area can contain up to 928 code words in a maximum of 90 rows, each of which can contain up to 30 code words.
QR Code Model 1	Convert the character string to the string in Shift-JIS, apply the escape sequence, and then encode the string based on the data type as shown below.
QR Code Model 2	Number: 0 to 9 Alphanumeric character: 0 to 9, A to Z, space, \$, %, *, +, -, ., /, : Kanji character: Shift-JIS value 8-bit, byte data: 0x00 to 0xff

2D-Code type	Description
MaxiCode Mode 2	Convert the character string to the string in UTF-8, apply the escape sequence, and then encode the string.
MaxiCode Mode 3	
MaxiCode Mode 4	
MaxiCode Mode 5	
MaxiCode Mode 6	In Modes 2 and 3, when the first piece of data is ()>\ x1e01\x1dyy (where yy is a two-digit number), this is processed as the message header, and the subsequent data is processed as the primary message. In other cases, from the first piece of data, data is processed as the primary message.
	In Mode 2, specify the primary message in the following format:
	Postal code (1- to 9-digit number) GS:(\x1d) ISO country code (1- to 3-digit number) GS:(\x1d) Service class code (1- to 3-digit number)
	In Mode 3, specify the primary message in the following format:
	Postal code (1 to 6 pieces of data convertible by Code Set A) GS:(\x1d) ISO country code (1- to 3-digit number) GS:(\x1d) Service class code (1- to 3-digit number)
GS1 DataBar Stacked	Convert the character string to the string in UTF-8, apply the escape sequence, and then encode the string.
GS1 DataBar Stacked Omnidirectional	Specify a 13-digit global trade item number (GTIN) not including an application identifier (AI) or a check digit.
GS1 DataBar Expanded Stacked	Convert the character string to the string in UTF-8, apply the escape sequence, and then encode the string.
	You can enclose an application identifier (AI) in parentheses. The parentheses are used as HRI print characters and are not encoded as data.
	To encode each of the following characters, specify two characters starting with the character "{":
	FNC1: {1
	(: {(
	): {)
Aztec Code Full-Range mode	After converting the character string to UTF-8, conduct the escape sequence and encode.
	Up to 3,067 characters of text, 3,832 numerical figures and 1,914 bytes of binary data can be specified.

4

2D-Code type	Description
Aztec Code Compact mode	After converting the character string to UTF-8, conduct the escape sequence and encode. Up to 89 characters of text, 110 numerical figures and 53 bytes of binary data can be specified.
DataMatrix square	After converting the character string to UTF-8, conduct the escape sequence and encode. The symbol is either a square ranging in size from 10 lines x 10 rows to 144 lines x 144 rows, or a rectangle comprising 8 lines, 12 lines or 16 lines. Up to 2,335 alphanumerical, 3,116 numerical figures and 1,556 bytes of binary data can be specified.
DataMatrix rectangle, 8 lines	
DataMatrix rectangle, 12 lines	
DataMatrix rectangle, 16 lines	

To specify binary data that cannot be represented by character strings, use the following escape sequences.

String	Description
\xnn	Control code
\\	Back slash

- type : Specifies the 2D-Code type.

Set value	2D-Code type
Builder.SYMBOL_PDF417_STANDARD	Standard PDF417
Builder.SYMBOL_PDF417_TRUNCATED	Truncated PDF417
Builder.SYMBOL_QRCODE_MODEL_1	QR Code Model 1
Builder.SYMBOL_QRCODE_MODEL_2	QR Code Model 2
Builder.SYMBOL_MAXICODE_MODE_2	MaxiCode Mode 2
Builder.SYMBOL_MAXICODE_MODE_3	MaxiCode Mode 3
Builder.SYMBOL_MAXICODE_MODE_4	MaxiCode Mode 4
Builder.SYMBOL_MAXICODE_MODE_5	MaxiCode Mode 5
Builder.SYMBOL_MAXICODE_MODE_6	MaxiCode Mode 6
Builder.SYMBOL_GS1_DATABAR_STACKED	GS1 DataBar Stacked
Builder.SYMBOL_GS1_DATABAR_STACKED_OMNIDIRECTIONAL	GS1 DataBar Stacked Omnidirectional
Builder.SYMBOL_GS1_DATABAR_EXPANDED_STACKED	GS1 DataBar Expanded Stacked
Builder.SYMBOL_AZTECCODE_FULLRANGE	Aztec Code Full-Range mode
Builder.SYMBOL_AZTECCODE_COMPACT	Aztec Code Compact mode
Builder.SYMBOL_DATAMATRIX_SQUARE	DataMatrix square
Builder.SYMBOL_DATAMATRIX_RECTANGLE_8	DataMatrix rectangle, 8 lines
Builder.SYMBOL_DATAMATRIX_RECTANGLE_12	DataMatrix rectangle, 12 lines
Builder.SYMBOL_DATAMATRIX_RECTANGLE_16	DataMatrix rectangle, 16 lines

- level : Specifies the error correction level.

Set value	Description
Builder.LEVEL_0	PDF417 error correction level 0
Builder.LEVEL_1	PDF417 error correction level 1
Builder.LEVEL_2	PDF417 error correction level 2
Builder.LEVEL_3	PDF417 error correction level 3
Builder.LEVEL_4	PDF417 error correction level 4
Builder.LEVEL_5	PDF417 error correction level 5
Builder.LEVEL_6	PDF417 error correction level 6
Builder.LEVEL_7	PDF417 error correction level 7
Builder.LEVEL_8	PDF417 error correction level 8
Builder.LEVEL_L	QR Code error correction level L
Builder.LEVEL_M	QR Code error correction level M
Builder.LEVEL_Q	QR Code error correction level Q
Builder.LEVEL_H	QR Code error correction level H
Builder.LEVEL_DEFAULT	Default level
5 to 95 integer	Aztec Code error correction level (percent unit)
Builder.PARAM_UNSPECIFIED	Retains the current setting.

> • Select the level according to the 2D-Code type.
> • For MaxiCode/two-dimensional GS1 DataBar/DataMatrix, select Builder.LEVEL_DEFAULT.

- width : Specifies the module width.

Set value	Description
Integer from 0 to 255	Module width
Builder.PARAM_UNSPECIFIED	Retains the current setting.

> MaxiCode is ignored.

- height : Specifies the module height.

Set value	Description
Integer from 0 to 255	Module height
Builder.PARAM_UNSPECIFIED	Retains the current setting.

> QR Code/MaxiCode/two-dimensional GS1 DataBar/Aztec Code/DataMatrix are ignored.

- size :   Specifies the 2D-Code maximum size.

Set value	Description
Integer from 0 to 65535	2D-Code maximum size
Builder.PARAM_UNSPECIFIED	Retains the current setting.

QR Code/MaxiCode/Aztec Code/DataMatrix are ignored.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To print 2D-Code:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addSymbol("ABCDE", Builder.SYMBOL_PDF417_STANDARD,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addSymbol("ABCDE", Builder.SYMBOL_QRCODE_MODEL_2,
            Builder.LEVEL_Q, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addSymbol("908063840\\x1d850\\x1d001\\x1d\\x04",
            Builder.SYMBOL_MAXICODE_MODE_2, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED);
    builder.addSymbol("0201234567890", Builder.SYMBOL_GS1_DATABAR_STACKED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addSymbol("0201234567890",
            Builder.SYMBOL_GS1_DATABAR_STACKED_OMNIDIRECTIONAL,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addSymbol("(01)02012345678903",
            Builder.SYMBOL_GS1_DATABAR_EXPANDED_STACKED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addPageBegin

Adds the switching to page mode to the command buffer. The page mode process starts.

> Use this API function with addPageEnd (p.84).

Syntax

```
public void addPageBegin() throws EposException
```

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To print the characters "ABCDE" in page mode:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addPageBegin();
    builder.addText("ABCDE");
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

4

addPageEnd

Adds the end of page mode to the command buffer. The page mode process ends.

> Use this API function with addPageBegin (p.83).

Syntax

```
public void addPageEnd() throws EposException
```

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To print the characters "ABCDE" in page mode:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addPageBegin();
    builder.addText("ABCDE");
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addPageArea

Adds the print area in page mode to the command buffer.
Specifies the print area in page mode (coordinates). After this API function, specify a print data API function such as the addText method.

⚠	Specify a print area to cover the content to be printed. If the print data extends beyond the print area, the print result will be such that the print data has been printed incompletely.
✎	Use this API function by inserting it between addPageBegin (p.83) and addPageEnd (p.84).

Syntax

```
public void addPageArea(int x, int y, int width
, int height) throws EposException
```

Parameter

- x : Specifies the origin of the horizontal axis (in dots). Specifies an integer from 0 to 65535. 0 is the left end of the printer's printable area.
- y : Specifies the origin of the vertical axis (in dots). Specifies an integer from 0 to 65535. 0 is the position in which no paper feed has been performed.
- width : Specifies the width of the print area (in dots). Specifies an integer from 0 to 65535.
- height : Specifies the height of the print area (in dots). Specifies an integer from 0 to 65535.

⚠	Determine the width and height of the print area according to the print direction setting. Otherwise, the print data might not be printed completely.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To specify the print area with the origin (100, 50), a width of 200 dots, and a height of 30 dots and print the characters "ABCDE":

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addPageBegin();
    builder.addPageArea(100, 50, 200, 30);
    builder.addText("ABCDE");
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addPageDirection

Adds the page mode print direction setting to the command buffer. Specifies the print direction in page mode. This function can be omitted if rotation is not required.

> Use this API function by inserting it between addPageBegin (p.83) and addPageEnd (p.84).

Syntax

```
public void addPageDirection(int dir)
  throws EposException
```

Parameter

- dir :   Specifies the print direction in page mode.

Set value	Description
Builder.DIRECTION_LEFT_TO_RIGHT (default)	Left to right (No rotation.Data is printed from the top left corner to the right.)
Builder.DIRECTION_BOTTOM_TO_TOP	Bottom to top (Counterclockwise rotation by 90 degrees. Data is printed from the bottom left corner to the top.)
Builder.DIRECTION_RIGHT_TO_LEFT	Right to left (Rotation by 180 degrees.Data is printed from the bottom right corner to the left.)
Builder.DIRECTION_TOP_TO_BOTTOM	Top to bottom (Clockwise rotation by 90 degrees. Data is printed from the top right corner to the bottom.)

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

4

Example

To print the characters "ABCDE" by rotating them 90 degrees clockwise:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addPageBegin();
    builder.addPageArea(100, 50, 30, 200);
    builder.addPageDirection(Builder.DIRECTION_TOP_TO_BOTTOM);
    builder.addText("ABCDE");
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addPagePosition

Adds the page mode print-position-set area to the command buffer.
Specifies the print start position (coordinates) in the area specified by the addPageArea method.

> Use this API function by inserting it between addPageBegin (p.83) and addPageEnd (p.84).

Syntax

```
public void addPagePosition(int x, int y)
  throws EposException
```

Parameter

- x :   Specifies the horizontal print position (in dots). Specifies an integer from 0 to 65535.
- y :   Specifies the vertical print position (in dots). Specifies an integer from 0 to 65535.

> Specify the print start position (coordinates) according to the content to be printed. Refer to the following.
> * To print a character string:
>   Specify the left end of the baseline for the first character. This can be omitted for left-aligned printing of standard-sized characters. To print double-sized height characters, specify a value equal to or greater than 42 for y.
> * To print a barcode:
>   Specify the bottom left of the symbol. And specify the barcode height for y.
> * To print a graphic/logo:
>   Specify the bottom left of the graphic data. And specify the graphic data height for y.
> * To print a 2D-Code:
>   Specify the top left of the symbol. This can be omitted when printing from the top left.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

4

Example

To specify (50,30) for the print start position in the area specified by the addPageArea method and print the characters "ABCDE":

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addPageBegin();
    builder.addPageArea(100, 50, 200, 100);
    builder.addPagePosition(50, 30);
    builder.addText("ABCDE");
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addPageLine

Adds line drawing in page mode to the command buffer. Draws a line in page mode.

⚠	Diagonal lines cannot be drawn.
✎	Use this API function by inserting it between addPageBegin (p.83) and addPageEnd (p.84).

Syntax

```
public void addPageLine
(int x1, int y1, int x2, int y2, int style)
 throws EposException
```

Parameter

- x1:   Specifies the horizontal start position of the line (in dots).
  Specifies an integer from 0 to 65535.
- y1:   Specifies the vertical start position of the line (in dots).
  Specifies an integer from 0 to 65535.
- x2 :   Specifies the horizontal end position of the line (in dots).
  Specifies an integer from 0 to 65535.
- y2 :   Specifies the vertical end position of the line (in dots).
  Specifies an integer from 0 to 65535.
- style :   Specifies the line type.

Set value	Description
Builder.LINE_THIN	Solid line: Thin
Builder.LINE_MEDIUM	Solid line: Medium
Builder.LINE_THICK	Solid line: Thick
Builder.LINE_THIN_DOUBLE	Double line: Thin
Builder.LINE_MEDIUM_DOUBLE	Double line: Medium
Builder.LINE_THICK_DOUBLE	Double line: Thick
Builder.PARAM_DEFAULT	Solid line: Thin

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

4

Example

To draw a thin solid line between the start position (100, 0) and the end position (500, 0):

```
try {
    Builder builder = new Builder("TM-P60", Builder.MODEL_ANK);
    builder.addPageBegin();
    builder.addPageLine(100, 0, 500, 0, Builder.LINE_THIN);
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addPageRectangle

Adds rectangle drawing in page mode to the command buffer. Draws a rectangle in page mode.

> Use this API function by inserting it between addPageBegin (p.83) and addPageEnd (p.84).

Syntax

```
public void addPageRectangle
(int x1, int y1, int x2, int y2, int style)
 throws EposException
```

Parameter

- x1:   Specifies the horizontal start position of the line (in dots).
               Specifies an integer from 0 to 65535.
- y1:   Specifies the vertical start position of the line (in dots).
               Specifies an integer from 0 to 65535.
- x2 :   Specifies the horizontal end position of the line (in dots).
               Specifies an integer from 0 to 65535.
- y2 :   Specifies the vertical end position of the line (in dots).
               Specifies an integer from 0 to 65535.
- style :   Specifies the line type.

Set value	Description
Builder.LINE_THIN	Solid line: Thin
Builder.LINE_MEDIUM	Solid line: Medium
Builder.LINE_THICK	Solid line: Thick
Builder.LINE_THIN_DOUBLE	Double line: Thin
Builder.LINE_MEDIUM_DOUBLE	Double line: Medium
Builder.LINE_THICK_DOUBLE	Double line: Thick
Builder.PARAM_DEFAULT	Solid line: Thin

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

4

Example

**To draw a rectangle with a thin solid line, with the start position (100, 0) and the end position (500, 200)
as its vertexes:**

```
try {
    Builder builder = new Builder("TM-P60", Builder.MODEL_ANK);
    builder.addPageBegin();
    builder.addPageRectangle(100, 0, 500, 200, Builder.LINE_THIN);
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addCut

Adds paper cut to the command buffer. Sets paper cut.

> ⚠️ Not available in page mode.

Syntax

```
public void addCut(int type) throws EposException
```

Parameter

- type :   Specifies the paper cut type.

Set value	Description
Builder.CUT_NO_FEED	Cut without feeding (The paper is cut without being fed.))
Builder.CUT_FEED	Feed cut (The paper is fed to the cut position and then is cut.)
Builder.CUT_RESERVE	Cut reservation (Printing continues until the cut position is reached, at which the paper is cut.)
Builder.PARAM_DEFAULT	Feed cut (The paper is fed to the cut position and then is cut.)

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To perform feed cut operation:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addCut(Builder.CUT_FEED);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addPulse

Adds the drawer kick to the command buffer. Sets the drawer kick.

> ⚠
> - Not available in page mode.
> - The drawer and the buzzer cannot be used together.

Syntax

```
public void addPulse(int drawer, int time)
  throws EposException
```

Parameter

- drawer : Specifies the drawer kick connector.

Set value	Description
Builder.DRAWER_1	Pin 2 of the drawer kick-out connector
Builder.DRAWER_2	Pin 5 of the drawer kick-out connector
Builder.PARAM_DEFAULT	Pin 2 of the drawer kick-out connector

- time : Specifies the ON time of the drawer kick signal.

Set value	Description
Builder.PULSE_100	100 ms
Builder.PULSE_200	200 ms
Builder.PULSE_300	300 ms
Builder.PULSE_400	400 ms
Builder.PULSE_500	500 ms
Builder.PARAM_DEFAULT	100 ms

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To send a 100msec pulse signal to the pin 2 of the drawer kick connector:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addPulse(Builder.DRAWER_1, Builder.PULSE_100);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addSound(For setting cycle buzzer)

Adds the turning on of the buzzer to the command buffer. Sets the buzzer.

> ⚠️
> - Not available in page mode.
> - The buzzer function and the drawer cannot be used together.
> - This API function cannot be used if the printer is not provided with the buzzer.

Syntax

```
public void addSound(int pattern, int repeat, int cycle)
  throws EposException
```

Parameter

- pattern :   Specifies the buzzer pattern.

Set value	Description
Builder.PATTERN_A	Pattern A
Builder.PATTERN_B	Pattern B
Builder.PATTERN_C	Pattern C
Builder.PATTERN_D	Pattern D
Builder.PATTERN_E	Pattern E
Builder.PATTERN_ERROR	Error sound pattern
Builder.PATTERN_PAPER_END	Pattern when there is no paper
Builder.PATTERN_1	Pattern 1
Builder.PATTERN_2	Pattern 2
Builder.PATTERN_3	Pattern 3
Builder.PATTERN_4	Pattern 4
Builder.PATTERN_5	Pattern 5
Builder.PATTERN_6	Pattern 6
Builder.PATTERN_7	Pattern 7
Builder.PATTERN_8	Pattern 8
Builder.PATTERN_9	Pattern 9
Builder.PATTERN_10	Pattern 10
Builder.PARAM_DEFAULT	Pattern A

- repeat :   Specifies the number of repeats.

Set value	Description
1 to 255	Number of repeats
Builder.PARAM_DEFAULT	One time

4

- cycle : This specifies the buzzer sounding cycle (in units of milliseconds).

Set value	Description
1000 to 25500	1000 to 25500 milliseconds
Builder.PARAM_DEFAULT	1000 milliseconds

"Pattern A to E"/ "Error sound pattern"/"Pattern when there is no paper" is disregarded.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

When sounding pattern 1 three times at 1,000 millisecond cycles

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addSound(Builder.PATTERN_1, 3, 1000);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addSound

Adds the turning on of the buzzer to the command buffer. Sets the buzzer.

	If you want to optionally set the buzzer sounding cycle (milliseconds), use addSound(For setting cycle buzzer) (p.97).

	• Not available in page mode. • The buzzer function and the drawer cannot be used together. • This API function cannot be used if the printer is not provided with the buzzer.

Syntax

```
public void addSound(int pattern, int repeat)
  throws EposException
```

Parameter

- pattern :   Specifies the buzzer pattern.

Set value	Description
Builder.PATTERN_A	Pattern A
Builder.PATTERN_B	Pattern B
Builder.PATTERN_C	Pattern C
Builder.PATTERN_D	Pattern D
Builder.PATTERN_E	Pattern E
Builder.PATTERN_ERROR	Error sound pattern
Builder.PATTERN_PAPER_END	Pattern when there is no paper
Builder.PATTERN_1	Pattern 1
Builder.PATTERN_2	Pattern 2
Builder.PATTERN_3	Pattern 3
Builder.PATTERN_4	Pattern 4
Builder.PATTERN_5	Pattern 5
Builder.PATTERN_6	Pattern 6
Builder.PATTERN_7	Pattern 7
Builder.PATTERN_8	Pattern 8
Builder.PATTERN_9	Pattern 9
Builder.PATTERN_10	Pattern 10
Builder.PARAM_DEFAULT	Pattern A

- repeat :   Specifies the number of repeats.

Set value	Description
1 to 255	Number of repeats
Builder.PARAM_DEFAULT	One time

4

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To repeat the sound pattern A three times:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addSound(Builder.PATTERN_A, 3);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addFeedPosition

Adds label / black mark paper feeding to the command buffer.

Syntax

```
public void addFeedPosition(int position)
  throws EposException
```

Parameter

- position :   Specifies the feed position.

Set value	Description
Builder.FEED_PEELING	Feeds to the peeling position.
Builder.FEED_CUTTING	Feeds to the cutting position.
Builder.FEED_CURRENT_TOF	Feeds to the top of the current label.
Builder.FEED_NEXT_TOF	Feeds to the top of the next label.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To feed a label paper to the peeling position:

```
try {
    Builder builder = new Builder("TM-P60II", Builder.MODEL_ANK);
    builder.addFeedPosition(Builder.FEED_PEELING);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

4

101

addLayout

Adds label / black mark paper layout information to the command buffer.

Syntax

```
public void addLayout(int type, int width, int height,
                      int marginTop, int marginBottom,
                      int offsetCut, int offsetLabel)
                      throws EposException
```

Parameter

- type :   Specifies the paper type.

Set value	Description
Builder.LAYOUT_RECEIPT	Receipt paper (no black mark)
Builder.LAYOUT_LABEL	Label paper (no black mark)
Builder.LAYOUT_LABEL_BM	Label paper (with black mark)
Builder.LAYOUT_RECEIPT_BM	Receipt paper (with black mark)

- width :   Specifies paper width (in units of 0.1mm). Specifies an integer from 1 to 10000.

- height :   Specifies the distance (in units of 0.1mm) from the standard printing position to the next standard printing position. Specifies an integer from 0 to 10000.
  If "0" is specified, the distance from the standard printing position to the next standard printing position is detected automatically.

- marginTop :   Specifies the distance (in units of 0.1mm) from the standard printing position to the top position. Specifies an integer from -9999 to 10000.

- marginBottom :Specifies the distance (in units of 0.1mm) from the standard eject position to the bottom edge of the printable area. Specifies an integer from -9999 to 10000.

- offsetCut :   Specifies the distance (in units of 0.1mm) from the standard eject position to the cutting position. Specifies an integer from -9999 to 10000.

- offsetLabel : Specifies the distance (in units of 0.1mm) from the standard eject position to the bottom edge of the label. Specifies an integer from 0 to 10000.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set 60mm label paper (black mark):

```
try {
    Builder builder = new Builder("TM-P60II", Builder.MODEL_ANK);
    builder.addLayout(Builder.PAPER_TYPE_LABEL_BM, 600, 0,
                      15, -15, 15, 0);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

Detailed description

❏ See below for the parameters that can be specified for each type of paper, and the positions for those parameters.

Mark	Parameter	Set value			
		Receipt	**Receipt (Black mark)**	**Label**	**Label (Black mark)**
A	width	1 to 10000	1 to 10000	1 to 10000	1 to 10000
B	height	0	0 to 10000	0 to 10000	0 to 10000
C	marginTop	0	-9999 to 10000	0 to 10000	-9999 to 10000
D	marginBottom	0	0	-9999 to 0	-9999 to 10000
E	offsetCut	0	-9999 to 10000	0 to 10000	0 to 10000
F	offsetLabel	0	0	0	0 to 10000

addCommand

Adds commands to the command buffer. Sends ESC/POS commands.

> ✏️ ESC/POS commands are not made public. For details, contact the dealer.

Syntax

```
public void addCommand(byte[] data) throws EposException
```

Parameter

- data : Specifies ESC/POS command as a binary data.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    byte[] data = null;
    ///Process///
    builder.addCommand(data);
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

Print class (Constructor)

Constructor for the Print class. Initializes a Print class instance.

> To use the log output function, use the Print class (Constructor) (For log output) (p.106).

Syntax

```
public Print()
```

Example

```
Print printer = new Print();

    ///Process///
```

Print class (Constructor) (For log output)

Constructor for the Print class. Initializes a Print class instance.
Use this constructor to use the log output function.

Syntax

```
public Print(Context context)
```

Parameter

- context :   Specifies the context of the application.

Example

```
import android.content.Context;

Print printer = new Print(getApplicationContext());

    ///Process///
```

openPrinter(For acquiring printer status)

This starts communications with the printer and monitoring of printer status.

	• Printer status is notified to the events registered in the print class. For details, see Automatic Acquisition of Printer Status (p.36). • If you want to stop monitoring of printer status, call closePrinter (p.111).
	• if communication with the printer is not required anymore, be sure to call closePrinter (p.111), closePrinter API, to end communication with the printer. • When you are opening the printer with another application via a Bluetooth connection, if you try to begin communication with this API, the process may not return.

Syntax

```
public void openPrinter
(int deviceType, String deviceName, int enabled,
 int interval) throws EposException
```

Parameter

- deviceType :   Specifies the type for the device to start communication.

Set value	Description
Print.DEVTYPE_TCP	Wi-Fi/Ethernet device
Print.DEVTYPE_BLUETOOTH	Bluetooth device

- deviceName : Specifies the identifier used for identification of the target device.
  Specifies the following for each device type:

Set value	Specified Value
Print.DEVTYPE_TCP	IP address (IPv4)
Print.DEVTYPE_BLUETOOTH	MAC address

- enabled :   This specifies whether printer status monitoring is enabled or disabled.

Set value	Specified Value
Print.TRUE	Enabled
Print.FALSE	Disabled
Print.PARAM_DEFAULT	Select default value (disabled)

- interval :   This specifies the interval (in units of milliseconds) for updating printer status.

Set value	Specified Value
1000 to 60000 integer	Interval for updating printer status (in units of milliseconds)
Print.PARAM_DEFAULT	Specify the default value (1000)

4

107

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_OPEN	Open processing failed.
ERR_ILLEGAL	An attempt was made to start communicating with the device with which communication had already started.
ERR_PROCESSING	Could not execute process.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

Case where printer status monitoring is enabled and communications are commenced using Wi-Fi/ Ethernet and a printer with an IP address of 192.168.192.168

```
Print printer = new Print();
try {
    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                        Print.PARAM_DEFAULT);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

openPrinter

Starts communication with the printer.

> If you want to automatically acquire the printer status, use openPrinter(For acquiring printer status) (p.107).

> • if communication with the printer is not required anymore, be sure to call closePrinter (p.111), closePrinter API, to end communication with the printer.
> • When you are opening the printer with another application via a Bluetooth connection, if you try to begin communication with this API, the process may not return.

Syntax

```
public void openPrinter
(int deviceType, String deviceName) throws EposException
```

Parameter

- deviceType : Specifies the type for the device to start communication.

Set value	Description
Print.DEVTYPE_TCP	Wi-Fi/Ethernet device
Print.DEVTYPE_BLUETOOTH	Bluetooth device

- deviceName : Specifies the identifier used for identification of the target device.
  Specifies the following for each device type:

Set value	Specified Value
Print.DEVTYPE_TCP	IP address (IPv4)
Print.DEVTYPE_BLUETOOTH	MAC address

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_OPEN	Open processing failed.
ERR_ILLEGAL	An attempt was made to start communicating with the device with which communication had already started.
ERR_PROCESSING	Could not execute process.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

4

Example

To start communication via Wi-Fi/Ethernet with the printer whose IP address is "192.168.192.168":

```
Print printer = new Print();
try {
    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

closePrinter

This ends communications with the printer and monitoring of printer status.

Syntax

```
public void closePrinter() throws EposException
```

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_ILLEGAL	This API was called when communication had not started yet.
ERR_PROCESSING	Could not execute process.
ERR_FAILURE	An unspecified error occurred.

Example

```
Print printer = new Print();
try {
    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");
    ///Process///
    printer.closePrinter();
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

4

sendData

Sends a print document created using the Builder class.

⚠	If you are using a Bluetooth connection, it may not be able to detect the offline status, and timeout errors may occur.
✎	If you want to acquire the battery status when sending a print document, use sendData(For acquiring battery status) (p.114).

Syntax

```
public void sendData(Builder builder, int timeout
, int[] status) throws EposException
```

Parameter

- builder :   Specifies a Builder class instance. For details on the Builder class, refer to Builder class (p.45).
- timeout :   Specifies the transmission/reception waiting timeout time.
                   Specifies an integer in the range 0-600000 (in milliseconds).
- status :   The printer status when command transmission ended is set. A combination of printer status settings is set. For details, refer to Printer Status List (p.43).

✎	In an exception occurs, use the getPrinterStatus (p.145) with exception processing to get the printer status.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_ILLEGAL	This API was called when communication had not started yet.
ERR_PROCESSING	Could not execute process.
ERR_TIMEOUT	Could not send all the data within the specified time.
ERR_CONNECT	Connection error occurred
ERR_MEMORY	Could not allocate memory.
ERR_OFF_LINE	The printer was offline.
ERR_FAILURE	An unspecified error occurred.

Example

To send a command to the printer by specifying 10 seconds for its timeout parameter:

```
Print printer = new Print();
int[] status = new int[1];
status[0] = 0;

try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addText("ABCDE");

    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");
    printer.sendData(builder, 10000, status);
    printer.closePrinter();
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
    status[0] = e.getPrinterStatus();
}
```

4

113

sendData(For acquiring battery status)

Sends a print document created using the Builder class.

> ⚠️ If you are using a Bluetooth connection, it may not be able to detect the offline status, and timeout errors may occur.

Syntax

```
public void sendData(Builder builder, int timeout
, int[] status, int[] battery) throws EposException
```

Parameter

- builder :   Specifies a Builder class instance. For details on the Builder class, refer to Builder class (p.45).

- timeout :   Specifies the transmission/reception waiting timeout time.
                  Specifies an integer in the range 0-600000 (in milliseconds).

- status :   The printer status when command transmission ended is set. A combination of printer status settings is set. For details, refer to Printer Status List (p.43).

- battery :   The battery status when command transmission ended is set.
                  For details, refer to Printer specifications (p.165).

> ✏️ If an exception occurs, use getPrinterStatus (p.145) with exception handling to acquire the printer status, and getBatteryStatus (p.146) for the battery status.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_ILLEGAL	This API was called when communication had not started yet.
ERR_PROCESSING	Could not execute process.
ERR_TIMEOUT	Could not send all the data within the specified time.
ERR_CONNECT	Connection error occurred
ERR_MEMORY	Could not allocate memory.
ERR_OFF_LINE	The printer was offline.
ERR_FAILURE	An unspecified error occurred.

Example

To send a command to the printer by specifying 10 seconds for its timeout parameter:

```
Print printer = new Print();
int[] status = new int[1];
int[] battery = new int[1];
status[0] = 0;
battery[0] = 0;

try {
    Builder builder = new Builder("TM-P60II", Builder.MODEL_ANK);
    builder.addText("ABCDE");

    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");
    printer.sendData(builder, 10000, status, battery);
    printer.closePrinter();
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
    status[0] = e.getPrinterStatus();
    battery[0] = e.getBatteryStatus();
}
```

4

115

setStatusChangeEventCallback

This registers the notification destination of printer status.

> • This API can be executed following execution of openPrinter(For acquiring printer status) (p.107).
> • When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten..

Syntax

```
public void setStatusChangeEventCallback
              (StatusChangeEventListener target)
```

Parameter

- target :   This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface StatusChangeEventListener
                    extends EventListener
```

Listener Registration Method

```
void onStatusChangeEvent(String deviceName, int status)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of printer status is set.
- status :   Printer status is set.

Example

```
public class SampleActivity extends Activity implements
                                 OnClickListener, StatusChangeEventListener {
    ///Process///
    private void onStatusChangeEvent(String deviceName, int status) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setStatusChangeEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

4

117

setOnlineEventCallback

This registers the notification destination of online events. This refers to events that are notified when printer status is online.

> • This API can be executed following execution of openPrinter(For acquiring printer status) (p.107).
> • When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten..

Syntax

```
public void setOnlineEventCallback
            (OnlineEventListener target)
```

Parameter

• target :   This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface OnlineEventListener extends
                    EventListener
```

Listener Registration Method

```
void onOnlineEvent(String deviceName)
```

Parameter

• deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of online event is set.

Example

```
public class SampleActivity extends Activity implements
                                OnClickListener, OnlineEventListener {
    ///Process///
    private void onOnlineEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setOnlineEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

4

setOfflineEventCallback

This registers the notification destination of offline events. This is the notification method when printer is offline concerning printer status.

> * This API can be executed following execution of openPrinter(For acquiring printer status) (p.107).
> * When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten..

Syntax

```
public void setOfflineEventCallback
            (OfflineEventListener target)
```

Parameter

* target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface OfflineEventListener extends
                    EventListener
```

Listener Registration Method

```
void onOfflineEvent(String deviceName)
```

Parameter

* deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of offline event is set.

Example

```
public class SampleActivity extends Activity implements
                                    OnClickListener, OfflineEventListener {
    ///Process///
    private void onOfflineEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setOfflineEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

4

setPowerOffEventCallback

This registers the notification destination of power off events. This refers to events that are notified when there is no response concerning printer status.

> • This API can be executed following execution of openPrinter(For acquiring printer status) (p.107).
> • When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten..

Syntax

```
public void setPowerOffEventCallback
              (PowerOffEventListener target)
```

Parameter

- target :   This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface PowerOffEventListener extends
                   EventListener
```

Listener Registration Method

```
void onPowerOffEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of power off event is set.

Example

```
public class SampleActivity extends Activity implements
                                OnClickListener, PowerOffEventListener {
    ///Process///
    private void onPowerOffEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setPowerOffEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

4

123

setCoverOkEventCallback

This registers the notification destination of cover close events. This refers to events that are notified when printer status indicates cover close.

> • This API can be executed following execution of openPrinter(For acquiring printer status) (p.107).
> • When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten..

Syntax

```
public void setCoverOkEventCallback
              (CoverOkEventListener target)
```

Parameter

- target :   This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface CoverOkEventListener extends
                    EventListener
```

Listener Registration Method

```
void onCoverOkEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of cover ok event is set.

Example

```
public class SampleActivity extends Activity implements
                                  OnClickListener, CoverOkEventListener {
    ///Process///
    private void onCoverOkEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setCoverOkEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

4

125

setCoverOpenEventCallback

This registers the notification destination of cover open events. This refers to events that are notified when the cover is open concerning printer status.

> - This API can be executed following execution of openPrinter(For acquiring printer status) (p.107).
> - When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten..

Syntax

```
public void setCoverOpenEventCallback
              (CoverOpenEventListener target)
```

Parameter

- target :   This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface CoverOpenEventListener extends
                      EventListener
```

Listener Registration Method

```
void onCoverOpenEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of cover open event is set.

Example

```
public class SampleActivity extends Activity implements
                                    OnClickListener, CoverOpenEventListener {
    ///Process///
    private void onCoverOpenEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setCoverOpenEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

4

setPaperOkEventCallback

This registers the notification destination of paper OK events. This refers to events that are notified when printer status indicates paper OK.

> • This API can be executed following execution of openPrinter(For acquiring printer status) (p.107).
> • When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten..

Syntax

```
public void setPaperOkEventCallback
            (PaperOkEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface PaperOkEventListener extends
                    EventListener
```

Listener Registration Method

```
void onPaperOkEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of paper ok event is set.

128

Example

```
public class SampleActivity extends Activity implements
                                    OnClickListener, PaperOkEventListener {
    ///Process///
    private void onPaperOkEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setPaperOkEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

4

setPaperNearEndEventCallback

This registers the notification destination of paper near end events. This refers to events that are notified when printer status indicates paper is near the end.

> • This API can be executed following execution of openPrinter(For acquiring printer status) (p.107).
> • When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten..

Syntax

```
public void setPaperNearEndEventCallback
            (PaperNearEndEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface PaperNearEndEventListener extends
                EventListener
```

Listener Registration Method

```
void onPaperNearEndEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of paper near end event is set.

Example

```
public class SampleActivity extends Activity implements
                                OnClickListener, PaperNearEndEventListener {
    ///Process///
    private void onPaperNearEndEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setPaperNearEndEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

4

setPaperEndEventCallback

This registers the notification destination of paper end events. This refers to events that are notified when printer status indicates there is no paper.

> • This API can be executed following execution of openPrinter(For acquiring printer status) (p.107).
> • When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten..

Syntax

```
public void setPaperEndEventCallback
            (PaperEndEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface PaperEndEventListener extends
                 EventListener
```

Listener Registration Method

```
void onPaperEndEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of paper end event is set.

Example

```
public class SampleActivity extends Activity implements
                                OnClickListener, PaperEndEventListener {
    ///Process///
    private void onPaperEndEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setPaperEndEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

4

setDrawerClosedEventCallback

This registers the notification destination of drawer closed events. This refers to events that are notified when printer status indicates the drawer is closed.

> • This API can be executed following execution of openPrinter(For acquiring printer status) (p.107).
> • When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten..

Syntax

```
public void setDrawerClosedEventCallback
              (DrawerClosedEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface DrawerClosedEventListener extends
                      EventListener
```

Listener Registration Method

```
void onDrawerClosedEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of drawer closed event is set.

Example

```
public class SampleActivity extends Activity implements
                                 OnClickListener, DrawerClosedEventListener {
    ///Process///
    private void onDrawerClosedEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setDrawerClosedEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

4

135

setDrawerOpenEventCallback

This registers the notification destination of drawer open events. This refers to events that are notified when printer status is drawer open.

> • This API can be executed following execution of openPrinter(For acquiring printer status) (p.107).
> • When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten..

Syntax

```
public void setDrawerOpenEventCallback
              (DrawerOpenEventListener target)
```

Parameter

- target :   This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface DrawerOpenEventListener extends
                    EventListener
```

Listener Registration Method

```
void onDrawerOpenEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of drawer open event is set.

Example

```
public class SampleActivity extends Activity implements
                                    OnClickListener, DrawerOpenEventListener {
    ///Process///
    private void onDrawerOpenEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setDrawerOpenEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }

    }
}
```

4

setBatteryLowEventCallback

This registers the notification destination of a battery low event. This refers to events that are notified when printer status is battery offline.

> • This API can be executed following execution of openPrinter(For acquiring printer status) (p.107).
> • When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten..

Syntax

```
public void setBatteryLowEventCallback
                (BatteryLowEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface BatteryLowEventListener extends
                    EventListener
```

Listener Registration Method

```
void onBatteryLowEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 format IP address / MAC address) of the device that performed battery low event notification is set.

Example

```
public class SampleActivity extends Activity implements
                                OnClickListener, BatteryLowEventListener {
    ///Process///
    private void onBatteryLowEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setBatteryLowEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

4

139

setBatteryOkEventCallback

This registers the notification destination of a battery OK event. This refers to events that are notified when the printer status recovers from offline due to remaining battery power.

> • This API can be executed following execution of openPrinter(For acquiring printer status) (p.107).
> • When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten..

Syntax

```
public void setBatteryOkEventCallback
            (BatteryOkEventListener target)
```

Parameter

- target :   This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface BatteryOkEventListener extends
                  EventListener
```

Listener Registration Method

```
void onBatteryOkEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 format IP address / MAC address) of the device that performed battery OK event notification is set.

Example

```
public class SampleActivity extends Activity implements
                              OnClickListener, BatteryOkEventListener {
    ///Process///
    private void onBatteryOkEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setBatteryOkEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

4

setBatteryStatusChangeEventCallback

This registers the notification destination of battery status.

> ✏️ • This API can be executed following execution of openPrinter(For acquiring printer status) (p.107).
> • When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten..

Syntax

```
public void setBatteryStatusChangeEventCallback
            (BatteryStatusChangeEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface BatteryStatusChangeEventListener
                    extends EventListener
```

Listener Registration Method

```
void onBatteryStatusChangeEvent
                    (String deviceName, int battery)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of battery status is set.
- battery : Battery status is set.

Example

```
public class SampleActivity extends Activity implements
                         OnClickListener, BatteryStatusChangeEventListener {
    ///Process///
    private void onBatteryStatusChangeEvent(String deviceName, int battery) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setBatteryStatusChangeEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

4

getErrorStatus

Acquire the error status from an exception.

Syntax

```
public int getErrorStatus()
```

Return value

Returns the error status set by the API in which an exception occurred.

Example

To acquire the error status from EposException.

```
try {
    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
    if (errStatus == EposException.ERR_OPEN) {
        ///Process///
    }
}
```

getPrinterStatus

Acquires the printer status from an exception that occurred in sendData (p.112).

Syntax

```
public int getPrinterStatus()
```

Return value

Returns the printer status. A combination of printer status settings is set.
For details, refer to Printer Status List (p.43).

Example

To acquire the printer status from EposException.

```
int[] printerStatus = new int[1];
printerStatus[0] = 0;
int timeout = 1000;

try {
    printer.sendData(builder, timeout, printerStatus);
} catch (EposException e) {
    int errSratus = e.getErrorStatus();
    if (errStatus == EposException.ERR_TIMEOUT) {
        printerStatus[0] = e.getPrinterStatus();
    }
}

if ((printerStatus[0] & Print.ST_PRINT_SUCCESS) == Print.ST_PRINT_SUCCESS)
{
    ///Process///
}
```

4

getBatteryStatus

Acquires the battery status from an exception that occurred in sendData(For acquiring battery status) (p.114).

Syntax

```
public int getBatteryStatus()
```

Return value

Returns the battery status. For details, refer to Printer specifications (p.165).

Example

To acquire the battery status from EposException.

```
int[] printerStatus = new int[1];
int[] batteryStatus = new int[1];
printerStatus[0] = 0;
batteryStatus[0] = 0;
int timeout = 1000;

try {
    printer.sendData(builder, timeout, printerStatus, batteryStatus);
} catch (EposException e) {
    int errSratus = e.getErrorStatus();
    if (errStatus == EposException.ERR_TIMEOUT) {
        printerStatus[0] = e.getPrinterStatus();
        batteryStatus[0] = e.getBatteryStatus();
    }
}

if ((printerStatus[0] & Print.ST_PRINT_SUCCESS) == Print.ST_PRINT_SUCCESS)
{
    ///Process///
}
```

Printer Search API

API to search for printers. The following classes are available.

❏ Finder class (p. 147)

❏ EpsonIoException class (p. 147)

Finder class

Class to search for printers. The following APIs are available.

API	Description	Page
start	Starts searching for printers.	148
stop	End communication with the printer.	149
getResult	Getting the printer search result.	150

EpsonIoException class

This class notifies you of the exception error value that occurred during the API calling of the Finder class and the Epsonio class (p.159).
The following APIs are available.

API	Description	Page
getStatus	Acquires an error value of an exception	151

start

Starts a search for printers of the specified device type.

<table>
<tr><td>⚠️</td><td>
• If you use this API, be sure to use stop (p.149) to stop the search.
• You cannot call this API when a printer search is already in progress.
</td></tr>
</table>

Syntax

```
public static synchronized void start
               (Context context, int deviceType,
                String findOption)
               throws EpsonIoException
```

Parameter

- context :   Set a Context class instance of caller.

  (Example: Set the Context acquired by getBaseContext() in Activity.)

- deviceType :   Specifies the device type to search for. The following values can be specified.

deviceType	Description
DevType.TCP	Searches for TM devices connected to the network
DevType.BLUETOOTH	Searches for Bluetooth devices that have a device class of Printer or Uncategorized.

- findOption :   Specifies the setting value when searching for a specific target device.

deviceType	Setting Value
DevType.TCP	The broadcast address to search for
DevType.BLUETOOTH	"null"

Exceptions

When processing fails, EpsonIoException is thrown with one of the following error values.

Error Value	Description
IoStatus.ERR_ILLEGAL	This API was called when a search was already in progress
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_PARAM	Invalid parameter was passed.
IoStatus.ERR_MEMORY	Could not allocate memory.
IoStatus.ERR_FAILURE	An unspecified error occurred.

stop

Stops the printer search.

Syntax

```
public static synchronized void stop()
                throws EpsonIoException
```

Exceptions

When processing fails, EpsonIoException is thrown with one of the following error values.

Error Value	Description
IoStatus.ERR_ILLEGAL	This API was called when a search was not in progress.
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_FAILURE	An unspecified error occurred.

4

getResult

Gets the printer search result until the time when this API was called.

> This API cannot acquire Bluetooth devices that are already open.

Syntax

```
public static synchronized final String[] getResult()
            throws EpsonIoException
```

Return value

The list of devices found during search is returned.

Identification information of the found devices is stored as a character string (String type) in the list.

The stored results differ depending on the type of device (deviceType).

deviceType	List to Acquire
DevType.TCP	List of IP addresses of printers
DevType.BLUETOOTH	List of MAC addresses of Bluetooth devices

Exceptions

When processing fails, EpsonIoException is thrown with one of the following error values.

Error Value	Description
IoStatus.ERR_ILLEGAL	This API was called when a search was not in progress.
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_MEMORY	Could not allocate memory.
IoStatus.ERR_FAILURE	An unspecified error occurred.

getStatus

Gets the error value of the exception.

Syntax

```
public int getStatus();
```

Return value

Returns the error value that is thrown with the exception. Error values are defined in the IoStatus class.

Error Value	Cause
IoStatus.ERR_PARAM	Invalid parameter was passed.
IoStatus.ERR_MEMORY	Could not allocate the necessary memory for processing.
IoStatus.ERR_ILLEGAL	Illegal method used.
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_FAILURE	An unspecified error occurred.

Log Setting API

Sets the log output. The following class is available.

❏ Log class ()

Log class

Sets the log output function.

API	Description	Page
setLogSettings	Sets the log output function.	

setLogSettings

Sets the log output function.

Syntax

```
public static void setLogSettings(Context context,
         int period, int enabled, String ipAddress,
         int port, int logSize, int logLevel)
          throws EposException
```

Parameter

- context :   Specifies the context of the application.
- period :   Specifies the method of setting the log output function.

Set value	Description
Log.LOG_TEMPORARY	The settings of this API are disabled when the application is ended.
Log.LOG_PERMANENT	The settings of this API are enabled even after the application is ended.

⚠ To specify period for the Log.LOG_PERMANENT, set the permissions for the application to access the storage.

- enabled : Specifies whether to enable the log output function and the log output destination.

Set value	Description
Log.LOG_DISABLE	Disables the log output function.
Log.LOG_STORAGE	Outputs log data to the device's storage.
Log.LOG_TCP	Outputs log data over TCP.

> ⚠ • To specify enabled for the Log.LOG_STORAGE, set the permissions for the application to access the storage.
> • To specify enabled for the Log.LOG_TCP, set the permissions for the application to access the network.

- ipAddress : Specifies the IPv4 IP address for TCP communication.

> ✎ If either of the following values is specified for period, "null" can be specified for this parameter.
> ∗ Log.LOG_DISABLE
> ∗ Log.LOG_STORAGE

- port : Specifies the port number for TCP communication. Specifies an integer from 0 to 65535.

> ✎ Even if either of the following values is specified for period, specify an integer within the range.
> ∗ Log.LOG_DISABLE
> ∗ Log.LOG_STORAGE

- logSize : Specifies the maximum size of log data that is saved on the device's storage. Specifies an integer from 1 to 50 (Unit:MB).

> ✎ Even if either of the following values is specified for period, specify an integer within the range.
> ∗ Log.LOG_DISABLE
> ∗ Log.LOG_TCP

- logLevel : Specifies the level of log data to be output.

Set value	Description
LOG_LOW	Low level

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_FAILURE	An unspecified error occurred.

4

Example

To output log data to port 8080 (IP address: 192.168.192.168) over TCP:

```
try {
   Log.setLogSettings(getApplicationContext(), Log.LOG_PERMANENT,
                    Log.LOG_TCP, "192.168.192.168", 8080, 10, Log.LOG_LOW);
} catch (EposException e) {
int errStatus = e.getErrorStatus();
    status[0] = e.getPrinterStatus();
}
```

To output log data to the device's storage:

```
try {
   Log.setLogSettings(getApplicationContext(), Log.LOG_PERMANENT,
                    Log.LOG_STORAGE, null,0, 10, Log.LOG_LOW);
} catch (EposException e) {
int errStatus = e.getErrorStatus();
    status[0] = e.getPrinterStatus();
}
```

To disable the log output function:

```
try {
   Log.setLogSettings(getApplicationContext(), Log.LOG_PERMANENT,
                    Log.LOG_DISABLE, null, 0, 10, Log.LOG_LOW);
} catch (EposException e) {
int errStatus = e.getErrorStatus();
    status[0] = e.getPrinterStatus();
}
```

Command Transmission/Reception

This chapter describes APIs for transmission and reception of commands (ESC/POS commands, etc.).

> A command transmission/reception API cannot be used with the Print class (p.47) of ePOS-Print API.

Programming

Programming Flow

Perform programming following this flow.

```
1. Printer search (p.30) *
   ❏ Starting the printer search (p.30)
   ❏ Getting the printer search result. (p.30)
   ❏ Stopping the printer search (p.30)
```

↓

```
2. Opening a Device Port (p.156)
```

↓

```
3. Sending Data (p.156)
```

↓

```
4. Receiving Data (p.157)
```

↓

```
5. Closing the Device Port (p.157)
```

*This is optional.

Opening a Device Port

Use the EpsonIo class's open (p.159) to open a device port. Please refer to the following code.

```
//Initialize the EpsonIo class
EpsonIo mPort = new EpsonIo();
int errStatus = IoStatus.SUCCESS;

//Open the device port
try {
    mPort.open(DevType.TCP, "192.168.192.168", null);
//Exception handling
} catch ( EpsonIoException e ) {
    errStatus = e.getStatus();
}
```

Sending Data

Use the EpsonIo class's write (p.162) to send data to the printer. Please refer to the following code.

Printing out "Hello, World!"

```
//Settings for sending
String str = "Hello, World!\r\n";
byte[] data = str.getBytes();
int offset = 0;
int size = data.length;
int timeout = 5000;
int sizeWritten = 0;
int errStatus = IoStatus.SUCCESS;

try {
//Send data
    sizeWritten = mPort.write(data, offset, size, timeout);
//Exception handling
} catch ( EpsonIoException e ) {
    errStatus = e.getStatus();
}
```

Receiving Data

Use the EpsonIo class's read (p.164) to receive data from the printer. Please refer to the following code.

```
//Settings for receiving
byte[] data = new byte[256];
int offset = 0;
int size = 256;
int timeout = 5000;
int sizeRead = 0;
int errStatus = IoStatus.SUCCESS;

//Receive data
try {
    sizeRead = mPort.read(data, offset, size, timeout);
//Exception handling
} catch ( EpsonIoException e ) {
    errStatus = e.getStatus();
}
```

Closing the Device Port

Use the EpsonIo class's close (p.161) to close the device port. Please refer to the following code.

```
int errStatus = IoStatus.SUCCESS;

//Close the device port
try {
    mPort.close();
//Exception handling
} catch ( EpsonIoException e ) {
    errStatus = e.getStatus();
}
```

5

Exception handling

A command transmission/reception API generates a propriety exception with an integer (int) type parameter when an error occurs and notify the calling side of such an error.

Steps for Handling

Use the EpsonIoException class's getStatus (p.151) to get the error value. Please refer to the following code.

```
String str = "Hello, World!\r\n";
byte[] data = str.getBytes();
int offset = 0;
int size = data.length;
int timeout = 5000;
int sizeWritten = 0;
int errStatus = IoStatus.SUCCESS;

try {
    sizeWritten = mPort.write(data, offset, size, timeout);
} catch ( EpsonIoException e ) {
//Get error value
    errStatus = e.getStatus();
}
```

List of Error Values

Error values are defined in the IoStatus class.

Error Value	Cause
IoStatus.ERR_PARAM	Invalid parameter was passed. <Example> • An invalid parameter such as null was passed. • A value outside the supported range was specified.
IoStatus.ERR_OPEN	Open processing failed.
IoStatus.ERR_CONNECT	Failed to connect to device. <Example> • Failed to send data to the target device for a reason other than a timeout. • Failed to receive data from the target device for a reason other than a timeout.
IoStatus.ERR_MEMORY	Could not allocate the necessary memory for processing.
IoStatus.ERR_ILLEGAL	Illegal method used. <Example> • The API for sending and receiving data was called when the device port was not open. • The printer search API was called again when a printer search was already in progress.
IoStatus.ERR_PROCESSING	Could not execute process. <Example> Could not get lock rights to the shared resource because the same process is currently being executed by another thread.
IoStatus.ERR_FAILURE	An unspecified error occurred.

Command Transmission/Reception API Reference

The following classes are available for command transmission/reception APIs:

EpsonIo class

Class to transmit and receive data. The following APIs are available.

API	Description	Page
open	Opens the device port.	159
close	Closes the device port.	161
write	Send data.	162
read	Receive data.	164

open

Opens the specified device port.

Syntax

```
public void open
              (int deviceType, String deviceName,
               String deviceSettings)
              throws EpsonIoException
```

Parameter

- deviceType :   Specifies the device type to open. The following values can be specified.

deviceType	Description
DevType.TCP	Specify this when the printer to be opened will connect with Wi-Fi/Ethernet.
DevType.BLUETOOTH	Specify this when the printer to be opened will connect with Bluetooth.

- deviceName : Specifies the identifier to locate the target device. The following values can be specified.

deviceType	Specified Value
DevType.TCP	IP address (IPv4)
DevType.BLUETOOTH	MAC address

- deviceSettings :
                 Specify "null".

159

Exceptions

When processing fails, EpsonIoException is thrown with one of the following error values.

Error Value	Description
IoStatus.ERR_OPEN	Open processing failed.
IoStatus.ERR_ILLEGAL	User attempted to open a device that is already open.
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_PARAM	Invalid parameter was passed.
IoStatus.ERR_MEMORY	Could not allocate memory.
IoStatus.ERR_FAILURE	An unspecified error occurred.

close

Closes the specified device port.

Syntax

```
public void close() throws EpsonIoException
```

Exceptions

When processing fails, EpsonIoException is thrown with one of the following error values.

Error Value	Description
IoStatus.ERR_ILLEGAL	This API was called when no device port was open.
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_FAILURE	An unspecified error occurred.

write

Sends data to a device port.

Syntax

```
public int write
            (byte[] data, int offset, int size,
             int timeout)
            throws EpsonIoException
```

Parameter

- data :   The sending data buffer. It stores data to be sent.
- offset :   Specifies the start position for sending data.
                  Please specify the offset value from the top of the sending data buffer.
- size :   Specifies the number of bytes to send.

> If "0" is specified for size, no data will be sent. In such a case, the return value will be "0".

- timeout :   Specifies the time in milliseconds to wait for sending to complete.
                  The maximum value that can be specified is 600000 (which equates to 10 minutes).

> - Take the transmission speed and volume of data to be sent into account when specifying the timeout value.
> - When the timeout value is too short, the sending process will still continue until all the data has been sent, while normal data sending is occurring, even if the timeout value is exceeded.
> - With a Bluetooth device, there is a chance that the sending process will be blocked. In such a case, processing will not complete even if the specified timeout value elapses.

Return value

Returns the number of bytes of data that were sent.

> The printer did not necessarily receive the amount of data that the return value shows.

> If the amount of time specified in timeout is exceeded, the returned return value is the number of bytes that were sent up to that point.

Exceptions

When processing fails, EpsonIoException is thrown with one of the following error values.

Error Value	Description
IoStatus.ERR_ILLEGAL	This API was called when no device port was open.
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_PARAM	Invalid parameter was passed.
IoStatus.ERR_CONNECT	Connection error occurred
IoStatus.ERR_MEMORY	Could not allocate memory.
IoStatus.ERR_FAILURE	An unspecified error occurred.

5

read

Receives data from a device port.

> ⚠ This API continues receiving until a receiving error occurs. However, if not even a single byte of data is received during the period specified in timeout, the process ends.

Syntax

```
public int read
            (byte[] data, int offset, int size,
             int timeout)
            throws EpsonIoException
```

Parameter

- data :   The receiving data buffer for storing received data.
- offset :   Specifies the point to start storing data in the receiving data buffer.
                  Please specify the offset value from the top of the receiving data buffer.
- size :   Specifies the number of bytes that can be received.

> ✎ If "0" is specified for size, no data will be received. In such a case, the return value will be "0".

- timeout :   Specifies the time in milliseconds to receive data. The maximum value that can be specified is 600000 (which equates to 10 minutes).

Return value

Returns the number of bytes that were received.

Exceptions

When processing fails, EpsonIoException is thrown with one of the following error values.

Error Value	Description
IoStatus.ERR_ILLEGAL	This API was called when no device port was open.
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_PARAM	Invalid parameter was passed.
IoStatus.ERR_CONNECT	Connection error occurred
IoStatus.ERR_MEMORY	Could not allocate memory.
IoStatus.ERR_FAILURE	An unspecified error occurred.

Appendix

Printer specifications

TM-T88V

	58mm	80mm
Interface	Ethernet, Wi-Fi	
Resolution	180 dpi x 180 dpi (W x H)	
Language	• ANK model • Japanese model • Chinese model • Taiwanese model • Korean model • South Asian model	
Print Width	360 dots	512 dots
Characters in a Line — Font A	ANK: 30 characters	ANK: 42 characters
Characters in a Line — Font B	ANK: 40 characters	ANK: 52 characters
Character Size — Font A	ANK: 12 dots x 24 dots (W x H)	
Character Size — Font B	ANK: 9 dots x 17 dots (W x H)	
Character Baseline — Font A	At the 21st dot from the top of the character	
Character Baseline — Font B	At the 16th dot from the top of the character	
Default Line Feed Space	30 dots	
Color Specification	First color	
Page Mode Default Area	360 dots x 831 dots (W x H)	512 dots x 831 dots (W x H)
Page Mode Maximum Area	360 dots x 1662 dots (W x H)	512 dots x 1662 dots (W x H)
Bar Code	UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 Databar Expanded	

	58mm	80mm
Two-Dimensional Code	PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked (Composite Symbology not supported)	
Paper Cut	Cut, Feed cut	
Drawer Kick-Out	Supported	
Buzzer	Optional	
Battery	Not Supported	

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
Builder Class			
Constructor	48	Constructor (For log output)	50
clearCommandBuffer	52	addTextAlign	53
addTextLineSpace	54	addTextRotate	55
addText	56	addTextLang	57
addTextFont	58	addTextSmooth	59
addTextDouble	60	addTextSize	61
addTextStyle	62	addTextPosition	64
addFeedUnit	65	addFeedLine	66
addImage(For multiple tone printing)	67	addImage	70
addLogo	72	addBarcode	73
addSymbol	78	addPageBegin	83
addPageEnd	84	addPageArea	85
addPageDirection	87	addPagePosition	89
addCut	95	addPulse	96
addSound(For setting cycle buzzer)	97	addSound	99
addCommand	104		
Print Class			
Constructor	105	Constructor (For log output)	106
openPrinter (For acquiring printer status)	107	openPrinter	109
closePrinter	111	sendData	112
setStatusChangeEventCallback	116	setOnlineEventCallback	118
setOfflineEventCallback	120	setPowerOffEventCallback	122
setCoverOkEventCallback	124	setCoverOpenEventCallback	126
setPaperOkEventCallback	128	setPaperNearEndEventCallback	130
setPaperEndEventCallback	132	setDrawerClosedEventCallback	134
setDrawerOpenEventCallback	136		
EposException Class			
getErrorStatus	144	getPrinterStatus	145

All the command transmission/reception APIs are supported.

TM-T70

	80mm
Interface	Ethernet, Wi-Fi
Resolution	180 dpi x 180 dpi (W x H)
Language	- ANK model - Japanese model - Chinese model - Taiwanese model - South Asian model
Print Width	512 dots
Characters in a Line Font A	ANK: 42 characters
Font B	ANK: 56 characters
Character Size Font A	ANK: 12 dots x 24 dots (W x H)
Font B	ANK: 9 dots x 17 dots (W x H)
Character Baseline Font A	At the 21st dot from the top of the chara
Font B	At the 15th dot from the top of the character
Default Line Feed Space	30 dots
Color Specification	First color
Page Mode Default Area	512 dots x 1662 dots (W x H)
Page Mode Maximum Area	512 dots x 1662 dots (W x H)
Bar Code	UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128
Two-Dimensional Code	QR Code
Paper Cut	Cut, Feed cut
Drawer Kick-Out	Supported
Buzzer	Not supported
Battery	Not supported

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
Builder Class			
Constructor	48	Constructor (For log output)	50
clearCommandBuffer	52	addTextAlign	53
addTextLineSpace	54	addTextRotate	55
addText	56	addTextLang	57
addTextFont	58	addTextSmooth	59
addTextDouble	60	addTextSize	61
addTextStyle	62	addTextPosition	64
addFeedUnit	65	addFeedLine	66
addImage(For multiple tone printing)	67	addImage	70
addLogo	72	addBarcode	73
addSymbol	78	addPageBegin	83
addPageEnd	84	addPageArea	85
addPageDirection	87	addPagePosition	89
addCut	95	addPulse	96
addCommand	104		
Print Class			
Constructor	105	Constructor (For log output)	106
openPrinter (For acquiring printer status)	107	openPrinter	109
closePrinter	111	sendData	112
setStatusChangeEventCallback	116	setOnlineEventCallback	118
setOfflineEventCallback	120	setPowerOffEventCallback	122
setCoverOkEventCallback	124	setCoverOpenEventCallback	126
setPaperOkEventCallback	128	setPaperNearEndEventCallback	130
setPaperEndEventCallback	132	setDrawerClosedEventCallback	134
setDrawerOpenEventCallback	136		
EposException Class			
getErrorStatus	144	getPrinterStatus	145

All the command transmission/reception APIs are supported.

TM-T70II

	80mm
Interface	Ethernet, Wi-Fi
Resolution	ANK: 180 dpi x 180 dpi (WxH) Other: 203 dpi x 203 dpi (W x H)
Language	• ANK model • Japanese model • Chinese model • Taiwanese model • Korean model • South Asian model
Print Width	ANK: 512 dots Other: 576 dots
Characters in a Line — Font A	ANK: 42 characters Other: 48 characters
Characters in a Line — Font B	ANK: 56 characters Other: 72 characters
Character Size — Font A	12 dots x 24 dots (W x H)
Character Size — Font B	9 dots x 17 dots (W x H)
Character Baseline — Font A	At the 21st dot from the top of the chara
Character Baseline — Font B	At the 15th dot from the top of the character
Default Line Feed Space	30 dots
Color Specification	First color
Page Mode Default Area	ANK: 512 dots x 1662 dots (WxH) Other: 576 dots x 1662 dots (WxH)
Page Mode Maximum Area	ANK: 512 dots x 1662 dots (WxH) Other: 576 dots x 1662 dots (WxH)
Bar Code	UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 Databar Expanded
Two-Dimensional Code	PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked

	80mm
Paper Cut	Cut, Feed cut
Drawer Kick-Out	Supported
Buzzer	Supported
Battery	Not supported

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
Builder Class			
Constructor	48	Constructor (For log output)	50
clearCommandBuffer	52	addTextAlign	53
addTextLineSpace	54	addTextRotate	55
addText	56	addTextLang	57
addTextFont	58	addTextSmooth	59
addTextDouble	60	addTextSize	61
addTextStyle	62	addTextPosition	64
addFeedUnit	65	addFeedLine	66
addImage(For multiple tone printing)	67	addImage	70
addLogo	72	addBarcode	73
addSymbol	78	addPageBegin	83
addPageEnd	84	addPageArea	85
addPageDirection	87	addPagePosition	89
addCut	95	addPulse	96
addSound(For setting cycle buzzer)	97	addSound	99
addCommand	104		
Print Class			
Constructor	105	Constructor (For log output)	106
openPrinter (For acquiring printer status)	107	openPrinter	109
closePrinter	111	sendData	112
setStatusChangeEventCallback	116	setOnlineEventCallback	118
setOfflineEventCallback	120	setPowerOffEventCallback	122
setCoverOkEventCallback	124	setCoverOpenEventCallback	126
setPaperOkEventCallback	128	setPaperNearEndEventCallback	130
setPaperEndEventCallback	132	setDrawerClosedEventCallback	134
setDrawerOpenEventCallback	136		
EposException Class			
getErrorStatus	144	getPrinterStatus	145

All the command transmission/reception APIs are supported.

TM-T90II

	80mm
Interface	Ethernet, Wi-Fi
Resolution	Japanese: 203 dpi x 203 dpi (W x H)
Language	• Japanese model
Print Width	576 dots
Characters in a Line	Font A	48 characters
	Font B	72 characters
	Font C	72 characters
Character Size	Font A	12 dots x 24 dots (W x H)
	Font B	9 dots x 17 dots (W x H)
	Font C	8 dots x 16 dots (W x H)
Character Baseline	Font A	At the 21st dot from the top of the chara
	Font B	At the 15th dot from the top of the character
	Font C	At the 15th dot from the top of the character
Default Line Feed Space	30 dots
Color Specification	First color
Page Mode Default Area	576 dots x 1662 dots (WxH)
Page Mode Maximum Area	576 dots x 1662 dots (WxH)
Bar Code	Codabar, Code39, ITF, JAN13(EAN), JAN8(EAN), UPC-A, UPC-E, Code93, Code128, GS1-128, GS1 DataBar Omni-directional, GS1 DataBar Truncated, GS1 DataBar Expanded, GS1 DataBar Limited
Two-Dimensional Code	PDF417, QRCode, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omni-directional, GS1 DataBar Expanded Stacked
Paper Cut	Cut, Feed cut
Drawer Kick-Out	Supported
Buzzer	Supported
Battery	Not supported

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
Builder Class			
Constructor	48	Constructor (For log output)	50
clearCommandBuffer	52	addTextAlign	53
addTextLineSpace	54	addTextRotate	55
addText	56	addTextLang	57
addTextFont	58	addTextSmooth	59
addTextDouble	60	addTextSize	61
addTextStyle	62	addTextPosition	64
addFeedUnit	65	addFeedLine	66
addImage(For multiple tone printing)	67	addImage	70
addLogo	72	addBarcode	73
addSymbol	78	addPageBegin	83
addPageEnd	84	addPageArea	85
addPageDirection	87	addPagePosition	89
addCut	95	addPulse	96
addCommand	104		
Print Class			
Constructor	105	Constructor (For log output)	106
openPrinter (For acquiring printer status)	107	openPrinter	109
closePrinter	111	sendData	112
setStatusChangeEventCallback	116	setOnlineEventCallback	118
setOfflineEventCallback	120	setPowerOffEventCallback	122
setCoverOkEventCallback	124	setCoverOpenEventCallback	126
setPaperOkEventCallback	128	setPaperNearEndEventCallback	130
setPaperEndEventCallback	132	setDrawerClosedEventCallback	134
setDrawerOpenEventCallback	136		
EposException Class			
getErrorStatus	144	getPrinterStatus	145

All the command transmission/reception APIs are supported.

TM-P60

	58mm	60mm
Interface	Wi-Fi, Bluetooth	
Resolution	203 dpi x 203 dpi (W x H)	
Language	ANK model	
Print Width	420 dots	432 dots
Characters in a Line — Font A	ANK: 35 characters	ANK: 36 characters
Font B	ANK: 42 characters	ANK: 43 characters
Font C	ANK: 52 characters	ANK: 54 characters
Character Size — Font A	ANK: 12 dots x 24 dots (W x H)	
Font B	ANK: 10 dots x 24 dots (W x H)	
Font C	ANK: 8 dots x 16 dots (W x H)	
Character Baseline — Font A	At the 21st dot from the top of the character	
Font B	At the 21st dot from the top of the character	
Font C	At the 15th dot from the top of the character	
Default Line Feed Space	30 dots	
Color Specification	First color	
Page Mode Default Area	420 dots x 1200 dots (W x H)	432 dots x 1200 dots (W x H)
Page Mode Maximum Area	420 dots x 1200 dots (W x H)	432 dots x 1200 dots (W x H)
Bar Code	UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR,CODE93, CODE128	
Two-Dimensional Code	Not supported	
Paper Cut	Cut, No cut	
Drawer Kick-Out	Not supported	
Buzzer	Supported	
Battery	Supported	

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
Builder Class			
Constructor	48	Constructor (For log output)	50
clearCommandBuffer	52	addTextAlign	53
addTextLineSpace	54	addTextRotate	55
addText	56	addTextLang	57
addTextFont	58	addTextSmooth	59
addTextDouble	60	addTextSize	61
addTextStyle	62	addTextPosition	64
addFeedUnit	65	addFeedLine	66
addFeedPosition*	101	addImage(For multiple tone printing)	67
addImage	70	addLogo	72
addBarcode	73	addPageBegin	73
addPageEnd	84	addPageArea	84
addPageDirection	87	addPagePosition	87
addPageLine	91	addPageRectangle	91
addCut	95	addLayout*	102
addCommand	95		
Print Class			
Constructor	105	Constructor (For log output)	106
openPrinter (For acquiring printer status)	107	openPrinter	109
closePrinter	111	sendData	112
sendData (For acquiring battery status)	114	setStatusChangeEventCallback	116
setOnlineEventCallback	118	setOfflineEventCallback	120
setPowerOffEventCallback	122	setCoverOkEventCallback	124
setCoverOpenEventCallback	126	setPaperOkEventCallback	128
setPaperNearEndEventCallback	130	setPaperEndEventCallback	132
setBatteryLowEventCallback	138	setBatteryOkEventCallback	140
setBatteryStatusChangeEventCallback	142		
EposException Class			
getErrorStatus	144	getPrinterStatus	145
getBatteryStatus	146		

*:Only for the peeler model.

> All the command transmission/reception APIs are supported.

Battery Status

Upper 8 bits

Battery Status	Cause
0x30	The AC adapter is connected
0x31	The AC adapter is not connected

Lower 8 bits

Battery Status	Cause
0x30	H level
0x31	M level
0x32	L level
0x33	S level
0x34	Battery not installed

> If 0x0000 is returned, the battery status cannot be acquired.

TM-P60II

	58mm	60mm
Interface	Wi-Fi, Bluetooth	
Resolution	203 dpi x 203 dpi (W x H)	
Country	• North America • Europe	
Print Width	420 dots	432 dots
Characters in a Line	Font A	ANK: 35 characters	ANK: 36 characters
	Font B	ANK: 42 characters	ANK: 43 characters
	Font C	ANK: 52 characters	ANK: 54 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H)	
	Font B	ANK: 10 dots x 24 dots (W x H)	
	Font C	ANK: 8 dots x 16 dots (W x H)	
Character Baseline	Font A	At the 21st dot from the top of the character	
	Font B	At the 16th dot from the top of the character	
	Font C	At the 15th dot from the top of the character	
Default Line Feed Space	30 dots	
Color Specification	First color	
Page Mode Default Area	420 dots x 831 dots (W x H)	576 dots x 831 dots (W x H)
Page Mode Maximum Area	420 dots x 1662 dots (W x H)	576 dots x 1662 dots (W x H)
Bar Code	UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF,CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 Databar Expanded	
Two-Dimensional Code	PDF417, QR Code, MaxiCode,GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked, Composite Symbology	
Paper Cut	Cut, Feed cut	
Drawer Kick-Out	Not supported	
Buzzer	Optional	
Battery	Supported	

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
Builder Class			
Constructor	48	Constructor (For log output)	50
clearCommandBuffer	52	addTextAlign	53
addTextLineSpace	54	addTextRotate	55
addText	56	addTextLang	57
addTextFont	58	addTextSmooth	59
addTextDouble	60	addTextSize	61
addTextStyle	62	addTextPosition	64
addFeedUnit	65	addFeedLine	66
addFeedPosition*	101	addImage(For multiple tone printing)	67
addImage	70	addLogo	72
addBarcode	73	addSymbol	78
addPageBegin	83	addPageEnd	84
addPageArea	85	addPageDirection	87
addPagePosition	89	addCut	95
addSound(For setting cycle buzzer)	97	addSound	99
addLayout*	102	addCommand	104
Print Class			
Constructor	105	Constructor (For log output)	106
openPrinter (For acquiring printer status)	107	openPrinter	111
closePrinter	111	sendData	114
sendData (For acquiring battery status)	114	setStatusChangeEventCallback	118
setOnlineEventCallback	118	setOfflineEventCallback	122
setPowerOffEventCallback	122	setCoverOkEventCallback	126
setCoverOpenEventCallback	126	setPaperOkEventCallback	130
setPaperNearEndEventCallback	130	setPaperEndEventCallback	138
setBatteryLowEventCallback	138	setBatteryOkEventCallback	142
setBatteryStatusChangeEventCallback	142		
EposException Class			
getErrorStatus	144	getPrinterStatus	145
getBatteryStatus	146		

*:Only for the peeler model.

All the command transmission/reception APIs are supported.

Battery Status

Upper 8 bits

Battery Status	Cause
0x30	The AC adapter is connected
0x31	The AC adapter is connected

Lower 8 bits

Battery Status	Cause
0x30	Battery amount 0 (real end)
0x31	Battery amount 1 (near end)
0x32	Battery amount 2
0x33	Battery amount 3
0x34	Battery amount 4
0x35	Battery amount 5
0x36	Battery amount 6

If 0x0000 is returned, the battery status cannot be acquired.

TM-U220

	76mm	69.5mm	57.5mm
Interface	Ethernet, Wi-Fi		
Resolution	160 dpi x 72 dpi (W x H)		
Language	• ANK model • Japanese model • Chinese model • Taiwanese model • Korean model • Thai model • South Asian model		
Print Width	400 or 385[*1] dots	360 dots	300 or 297[*1] dots
Characters in a Line — Font A	ANK: 40 characters	ANK: 36 characters	ANK: 30 characters
Characters in a Line — Font B	ANK: 33 characters	ANK: 30 characters	ANK: 25 characters
Character Size — Font A	ANK: 7 dots x 9 dots (W x H)		
Character Size — Font B	ANK: 9 dots x 9 dots (W x H)		
Character Baseline — Font A	-		
Character Baseline — Font B	-		
Default Line Feed Space	12 dots		
Color Specification	First color		
Page Mode Default Area	-		
Page Mode Maximum Area	-		
Bar Code	Not supported		
Two-Dimensional Code	Not supported		
Paper Cut	Cut, No cut		
Drawer Kick-Out	Supported		
Buzzer	Not supported		
Battery	Not supported		

*1: DipSW2-1 = ON

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
Builder Class			
Constructor	48	Constructor (For log output)	50
clearCommandBuffer	52	addTextAlign	53
addTextLineSpace	54	addTextRotate	55
addText	56	addTextLang	57
addTextFont	58	addTextStyle	62
addFeedUnit	65	addFeedLine	66
addImage(For multiple tone printing)	67	addImage	70
addCut	95	addPulse	96
addCommand	104		
Print Class			
Constructor	105	Constructor (For log output)	106
openPrinter (For acquiring printer status)	107	openPrinter	109
closePrinter	111	sendData	112
setStatusChangeEventCallback	116	setOnlineEventCallback	118
setOfflineEventCallback	120	setPowerOffEventCallback	122
setCoverOkEventCallback	124	setCoverOpenEventCallback	126
setPaperOkEventCallback	128	setPaperNearEndEventCallback	130
setPaperEndEventCallback	132	setDrawerClosedEventCallback	134
setDrawerOpenEventCallback	136		
EposException Class			
getErrorStatus	144	getPrinterStatus	145

> All the command transmission/reception APIs are supported.

TM-T20

		58mm	80mm
Interface		Ethernet	
Resolution		203 dpi x 203 dpi (W x H)	
Language		• ANK model • Japanese model	
Print Width		420 dots	576 dots
Characters in a Line	Font A	ANK: 35 characters	ANK: 48 characters
	Font B	ANK: 46 characters	ANK: 64 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H)	
	Font B	ANK: 9 dots x 17 dots (W x H)	
Character Baseline	Font A	At the 21st dot from the top of the character	
	Font B	At the 16th dot from the top of the character	
Default Line Feed Space		30 dots	
Color Specification		First color	
Page Mode Default Area		420 dots x 831 dots (W x H)	576 dots x 831 dots (W x H)
Page Mode Maximum Area		420 dots x 1662 dots (W x H)	576 dots x 1662 dots (W x H)
Bar Code		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF,CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 Databar Expanded	
Two-Dimensional Code		PDF417, QR Code, MaxiCode,GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked, Composite Symbology	
Paper Cut		Cut, Feed cut	
Drawer Kick-Out		Supported	
Buzzer		Not supported	
Battery		Not supported	

List of Supported ePOS-Print APIs

All the command transmission/reception APIs are supported.

TM-T81II

	80mm
Interface	Ethernet
Resolution	203 dpi x 203 dpi (W x H)
Language	Simplified Chinese model
Print Width	576 dots
Characters in a Line — Font A	ANK: 48 characters
Font B	ANK: 64 characters
Character Size — Font A	ANK: 12 dots x 24 dots (W x H)
Font B	ANK: 9 dots x 17 dots (W x H)
Character Baseline — Font A	At the 21st dot from the top of the character
Font B	At the 16th dot from the top of the character
Default Line Feed Space	30 dots
Color Specification	First color
Page Mode Default Area	576 dots x 831 dots (W x H)
Page Mode Maximum Area	576 dots x 1662 dots (W x H)
Bar Code	UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF,CODABAR, CODE93, CODE128
Two-Dimensional Code	PDF417, QR Code
Paper Cut	Cut, Feed cut
Drawer Kick-Out	Supported
Buzzer	Not supported
Battery	Not supported

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
Builder Class			
Constructor	48	Constructor (For log output)	50
clearCommandBuffer	52	addTextAlign	53
addTextLineSpace	54	addTextRotate	55
addText	56	addTextLang	57
addTextFont	58	addTextSmooth	59
addTextDouble	60	addTextSize	61
addTextStyle	62	addTextPosition	64
addFeedUnit	65	addFeedLine	66
addImage(For multiple tone printing)	67	addImage	70
addLogo	72	addBarcode	73
addSymbol	78	addPageBegin	83
addPageEnd	84	addPageArea	85
addPageDirection	87	addPagePosition	89
addCut	95	addPulse	96
addCommand	104		
Print Class			
Constructor	105	Constructor (For log output)	106
openPrinter (For acquiring printer status)	107	openPrinter	109
closePrinter	111	sendData	112
setStatusChangeEventCallback	116	setOnlineEventCallback	118
setOfflineEventCallback	120	setPowerOffEventCallback	122
setCoverOkEventCallback	124	setCoverOpenEventCallback	126
setPaperOkEventCallback	128	setPaperNearEndEventCallback	130
setPaperEndEventCallback	132	setDrawerClosedEventCallback	134
setDrawerOpenEventCallback	136		
EposException Class			
getErrorStatus	144	getPrinterStatus	145

All the command transmission/reception APIs are supported.

TM-T82

		58mm	80mm
Interface		Ethernet	
Resolution		203 dpi x 203 dpi (W x H)	
Language		• Chinese model • South Asian model	
Print Width		420 dots	576 dots
Characters in a Line	Font A	ANK: 35 characters	ANK: 48 characters
	Font B	ANK: 46 characters	ANK: 64 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H)	
	Font B	ANK: 9 dots x 17 dots (W x H)	
Character Baseline	Font A	At the 21st dot from the top of the character	
	Font B	At the 16th dot from the top of the character	
Default Line Feed Space		30 dots	
Color Specification		First color	
Page Mode Default Area		420 dots x 831 dots (W x H)	576 dots x 831 dots (W x H)
Page Mode Maximum Area		420 dots x 1662 dots (W x H)	576 dots x 1662 dots (W x H)
Bar Code		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF,CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 Databar Expanded	
Two-Dimensional Code		PDF417, QR Code, MaxiCode,GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked, Composite Symbology	
Paper Cut		Cut, Feed cut	
Drawer Kick-Out		Supported	
Buzzer		Optional	
Battery		Not supported	

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
Builder Class			
Constructor	48	Constructor (For log output)	50
clearCommandBuffer	52	addTextAlign	53
addTextLineSpace	54	addTextRotate	55
addText	56	addTextLang	57
addTextFont	58	addTextSmooth	59
addTextDouble	60	addTextSize	61
addTextStyle	62	addTextPosition	64
addFeedUnit	65	addFeedLine	66
addImage(For multiple tone printing)	67	addImage	70
addLogo	72	addBarcode	73
addSymbol	78	addPageBegin	83
addPageEnd	84	addPageArea	85
addPageDirection	87	addPagePosition	89
addCut	95	addPulse	96
addSound(For setting cycle buzzer)	97	addSound	99
addCommand	104		
Print Class			
Constructor	105	Constructor (For log output)	106
openPrinter (For acquiring printer status)	107	openPrinter	109
closePrinter	111	sendData	112
setStatusChangeEventCallback	116	setOnlineEventCallback	118
setOfflineEventCallback	120	setPowerOffEventCallback	122
setCoverOkEventCallback	124	setCoverOpenEventCallback	126
setPaperOkEventCallback	128	setPaperNearEndEventCallback	130
setPaperEndEventCallback	132	setDrawerClosedEventCallback	134
setDrawerOpenEventCallback	136		
EposException Class			
getErrorStatus	144	getPrinterStatus	145

All the command transmission/reception APIs are supported.

TM-T82II

		58mm	80mm
Interface		Ethernet	
Resolution		203 dpi x 203 dpi (W x H)	
Language		• ANK model • Chinese model • Taiwanese model • South Asian model	
Print Width		420 dots	576 dots
Characters in a Line	Font A	ANK: 35 characters	ANK: 48 characters
	Font B	ANK: 46 characters	ANK: 64 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H)	
	Font B	ANK: 9 dots x 17 dots (W x H)	
Character Baseline	Font A	At the 21st dot from the top of the character	
	Font B	At the 16th dot from the top of the character	
Default Line Feed Space		30 dots	
Color Specification		First color	
Page Mode Default Area		420 dots x 831 dots (W x H)	576 dots x 831 dots (W x H)
Page Mode Maximum Area		420 dots x 1662 dots (W x H)	576 dots x 1662 dots (W x H)
Bar Code		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF,CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 Databar Expanded	
Two-Dimensional Code		PDF417, QR Code, MaxiCode,GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked, Composite Symbology	
Paper Cut		Cut, Feed cut	
Drawer Kick-Out		Supported	
Buzzer		Optional	
Battery		Not supported	

List of Supported ePOS-Print APIs

All the command transmission/reception APIs are supported.